No input authoriz...

Generate Collection | Print

L5: Entry 4 of 10 File: USPT Jun 2, 1998

DOCUMENT-IDENTIFIER: US 5761684 A
TITLE: Method and reusable object for scheduling script execution in a compound document

DATE FILED (1):
19950530

Abstract Text (1):
A reusable script execution scheduling part for compound documents in a document-centric processing environment. Document-centric computing environments having architectures similar to OpenDoc.TM. include a technique for executing scripts to interact with the compound document content. The CHRON part of this invention includes embedded objects for defining scheduled execution times for one or more scripts that may be opened within the compound document to provide a view of its contents. Either the ScheduleTime element or the ScriptEvent element of the CHRON part may be opened and edited in place. The reusable CHRON part sets up event scheduling with the operating system so that specified scripts are run according to specified ScheduleTimes.

Brief Summary Text (3):
This invention relates generally to event scheduling in object-oriented application systems and specifically to a reusable compound document script scheduling part that provides class methods for the scheduled execution of embedded scripts in an object-oriented compound document processing system.

Brief Summary Text (11):
With the development of the CORBA specification and the related Interface Definition Language (IDL) specification, several new object-oriented system environments have been developed by practitioners in the art, including a class of environments herein denominated "document-centric" processing system. The emergence of document-centric computing technology represents a major event in the transition to object-oriented technology from process-oriented technology. The document-centric computing art now includes at least three commercially-available compound document architectures. These include Microsoft's Object Linking and Embedding (OLE.TM.) System, the Taligent.TM. consortium's compound document framework, and the Component Integration Laboratory (CI Labs) OpenDoc.TM. compound document application architecture. CI Labs is an independent industry consortion responsible for the OpenDoc architecture. Reference is made to the OpenDoc white paper (OpenDoc: Shaping Tomorrows Software, Apple Computer, Inc., Cupertino, Calif., 1993) for an introduction to the OpenDoc architecture.

Brief Summary Text (12):
Major elements of the OpenDoc architecture include objects for "documents," "parts," "part handlers," and "frames." The OpenDoc "document" is a "compound document" and is fundamentally different from the usual meaning of the term document. A compound document is no longer a single block of content bound to a single application but is instead composed of smaller blocks of content herein denominated "parts." Parts are the fundamental building blocks of the OpenDoc architecture, replacing monolithic applications with smaller units of content dynamically bound to related functionality. OpenDoc parts each contain data. For example, text parts contain characters, graphics parts contain lines and shapes, spreadsheet parts contain spreadsheet cells with formulas, and video parts contain digitized video. The particular data type within each part is defined by the part developer and is denominated the part's "intrinsic content." In addition to intrinsic content, a part may contain other parts, each having its own intrinsic content. Every compound

document has a single "root part" at the top level into which all other parts are embedded. Usually if a part can contain one type of part, it may contain all conforming types of parts, including OpenDoc parts yet to be developed.

Brief Summary Text (13):
Parts may also be viewed as the boundaries at which one kind of content in a document ends and another begins. Each part of a document has its own content model, which is the model of objects and operations that is presented to the user. The content model changes at the "frame" between parts in a compound document. Frames are areas of the display that represent a part but also serve as a handle permitting manipulation of the part as a whole as well as permitting the user to see and edit the intrinsic contents of a part. A frame is much more than a standard application window, which is only visible when the part is being viewed or edited. A frame is persistent and remains visible when opened into a window. When the window is closed, the part returns to the representation from which it was opened and remains as a persistent element of the compound document.

Brief Summary Text (14):
In the OpenDoc architecture, "part handlers" are the rough equivalent of application programs. Part handlers include part editors and part viewers and may be responsible for displaying the part both on the screen and for printing purposes, editing the part, and storage management (both persistent and runtime) for the part. The part handler must read the part from persistent storage into main memory, manage the runtime storage associated with the part and write the part back out to persistent storage when appropriate. Part handlers are dynamically linked into the runtime world of the compound document depending on the part types that appear in the compound document.

Brief Summary Text (16):
The OpenDoc architecture storage model is based on the Apple Computer, Inc. Bento.TM. Object Container System (OCS) that functions as the reference file storage system, the object container for parts placed on a clipboard, and the container for transporting documents across platforms. The OpenDoc architecture also provides the Open Scripting Architecture (OSA) for scripting languages such as OREXX, AppleScript, ScriptX, and OLE Automation. OSA includes script typing that permits identification of the correct scripting engine for script processing and standardization of script semantic messages according to the Open Events form required by OSA. Scripts may become a part of the compound document, either attached or embedded as necessary. They may also operate as control structures, as front- or back-ends to operations or as instructions for use embedded into a compound document for transfer across system boundaries. Parts that are OSA-scriptable must register their capabilities by declaring the event suites that they support. Scripts are then built for a particular set of event suites and those scripts then may run on all parts that support those particular suites.

Brief Summary Text (17):
According to Robert Orfali et al. ("Special Report: Intergalactic Client/Server Computing," Byte, pp. 108-122, April 1995), the commercial alliances arising from the OMG, including CI Labs, may eventually lead to an object infrastructure that can meet the demands of the "intergalactic" client-server era (the "third wave" of computer network technology). However, Orfali et al. observe that the present state of the art is not yet capable of providing the object services necessary for the "third wave" era and cite problems such as transactions, locking, life cycle, naming, and persistence. Moreover, not all document-centric architectures (e.g., OLE) are yet available in the CORBA-compliant form required for distributed systems.

Brief Summary Text (18):
Another problem felt in the compound document art is the lack of any useful method for scheduling a point in time for executing an OSA script in a compound document. This particular problem results in part from the power of the OSA, with which scripts may be either attached or embedded in compound documents or in other parts as desired and thereafter carried along across boundaries throughout a distributed system with the parent part or document. Scripts may be executed either in response to a user command or in response to a message from another object, but execution of a script at a predetermined date and time is more problematic. The event scheduling element of the operating system must first create and store a persistent log entry specifying the date and time for execution of the script in the system event log. If a special-purpose application program is used to create the necessary log entry, the

resulting application object does not provide for "reuse," thereby increasing programmer workload and system complexity. If the necessary date and time commands are included in the script itself, then the script cannot be "reused" and must be revised for each desired execution data and time. That is, the usual process-centric solutions to script-execution scheduling are not effective in the document-centric system model, and no user alternative techniques have been suggested by practitioners in the art until now.

Brief Summary Text (21):
In U.S. Pat. No. 4,937,743, Rassman et al. disclose a project resource scheduling GUI system that provides automatic conflict resolution, periodic monitoring and alarm-triggering of scripts. Again, although Rassman et al. consider the dynamic management of a plurality of resources, they neither consider nor suggest reusable objects for object-oriented processing in a document-centric system architecture. In U.S. Pat. No. 5,063,523, Vrenjack discloses a data communication network management system that permits a user to establish rules for pattern-matching to selected attributes of incoming events, such as alarms, from network objects. Although Vrenjack describes a process-centric system including the well-known concept of an event that can trigger a script, where the event may also be date or time based, he neither considers nor suggests any method for providing a reusable compound document script scheduling part suitable for document-centric applications.

Brief Summary Text (22):
Accordingly, there is still an unmet need for a class type or method to run scripts at predetermined times in a document-centric computing environment such as the OpenDoc architecture. Such a method must operate within the context and limitations of the compound document without unnecessarily avoiding the advantageous features of the document-centric computing environment. The related unresolved problems and deficiencies are clearly felt in the art and are solved by this invention in the manner described below.

Brief Summary Text (24):
This invention solved the script execution scheduling problem by introducing a new class type, herein denominated the CHRON type, for scheduling script execution in a compound document. Because each "instance" of the CHRON class is an object, it provides the reusability and extensibility features necessary for proper integration into a document-centric architecture such as the OpenDoc architecture. The CHRON part defines the point in time at which a script is to be executed and includes one or more embedded script parts that specify the scripts to be executed, one or more embedded times/date parts that specify when the scripts are to be executed and may also contain embedded tabular specifications of relationships between time parts and script parts. The CHRON part is represented to the user when minimized by an icon that displays the predetermined script execution time.

Brief Summary Text (25):
It is an object of this invention to provide a reusable compound document script scheduling part for automatic script execution at predetermined times. It is a feature of the CHRON part of this invention that it is both reusable and extensible in accordance with object-oriented programming architectural requirements. It is an advantage of the CHRON part of this invention that it may be manipulated by the user through a Graphical User Interface (GUI) to quickly and easily revise either the designated scripts or the time/date parts that specify when the scripts are executed.

Drawing Description Text (7):
FIG. 5 shows an example of a compound document in accordance with the OpenDoc.TM. Architecture from the prior art;

Drawing Description Text (9):
FIG. 7 is a functional block diagram of an illustrative embodiment of a compound document including the CHRON part of this invention;

Detailed Description Text (2):
The class type and object of this invention may be better understood in view of a brief discussion of the object-oriented programming art in general and compound document architecture in particular. With the advent of object-oriented programming techniques leading to compound document architectures such as OpenDoc and OLE and the document framework in Taligent, the computing model for the user has evolved from a "tool-centric" to a "document-centric" model. In the document-centric model

of computing, the user may focus on the document desired rather than on the output of some specific tool or application program. To the user, a document is now composed of several "parts" that are each embedded and integrated into a "document." The parts are "class types" or objects that may now be completely incorporated into the document. Part usefulness depends only on availability of the necessary part handler application (e.g., part editors and part viewers).

Detailed Description Text (3):
Compound Document Architecture In General

Detailed Description Text (4):
The generic document-centric system providing for the embedding of objects into a document is embodied in an object-oriented system such as the OpenDoc or Taligent document framework. The fundamental concept of the compound document is that it can hold different kinds of data in the form of individual document parts and that each kind of part is handled by an independent application or part handler. Each part handler independently understands its own intrinsic content and need not know anything about the intrinsic content of other parts embedded therein. The integration and cooperation of these parts is accomplished by the compound document system architecture, policies and protocols.

Detailed Description Text (5):
FIG. 1 shows the class library of objects (SOM or CORBA objects) employed by the OpenDoc system to realize the basic interfaces necessary in a compound document architecture. FIG. 1 also shows the inheritance relationships of the OpenDoc class library known in the art. FIG. 2 shows the runtime relationships of the primary OpenDoc object type, which is more meaningful to the system developer. FIGS. 1 and 2 exemplify one embodiment (the OpenDoc system) of the general compound document architecture, which is now discussed.

Detailed Description Text (6):
A compound document consists of parts embedded in a document container. The content of any particular part is not limited because each part is isolated from the other by the document architecture. The only practical limitation on content is the availability of a part handler, either editor or viewer. As new part handlers are created, they may be immediately used to manage the appropriate parts. The user may rely on a clipboard and on drag-and-drop mechanisms within the document. Part handlers may be changed or replaced to fit specific user requirements, which may be satisfied independently of existing documents because these do not depend on any specific part handler. The different parts are isolated by using linkages for data transfer and navigation, which may be both internal to the compound document (such as hypertext-type links) and external to the compound document (for data transfer and navigation).

Detailed Description Text (7):
The fundamental problems faced in a compound document system include data exchange, part registration, part-to-part communication, document layout, user interface control, events and messages, document versioning and document storage. Many compound documents are built-up by the movement of data from one document (or the desktop) to another either through a cut-and-paste or a drag-and-drop operation by the user. These interfaces determine how data is moved from one document to another and must therefore define how data is added and removed from the clipboard. The interfaces must also transfer type and attribute information so that the proper part handler application program can be retrieved from the part handler database and launched. Because the clipboard must transfer complex data-objects, an in-memory object container is also required. The storage format standard for clipboard data in OpenDoc is the Bento container shown in FIG. 3.

Detailed Description Text (9):
Part-to-part communication requires an interface between parts and between the containing part and its embedded subparts. This interface must provide access to internal part data, pass to the parts commands such as "save your data," "externalize yourself," and the like, and must pass requests to accept or release control of some global resource such as the menu bar or input focus.

Detailed Description Text (10):
With a number of embedded parts sharing space on the screen and in a printed document, central coordination is required over display layout. The display layout controls operate with document window geometry and the geometry of embedded parts,

which contains information about the size, shape, position, clip extents and transformations of the frames and windows together with the means for establishing and manipulating them. A document layout interface must handle the negotiation for space resources between the part handler and the container. It is used when the part must grow or shrink in size in response to user interactions such as picture editing or data changes arising from a link update of embedded data. The part container establishes the policy for document layout and has final authority therefor. Such negotiations may result in the part being placed on a new page, or being split into more than one piece or in the rearrangement of the document to accommodate a new part size or even in refusal of authority for the part to change size.

Detailed Description Text (13):
Document collaboration and versioning interfaces manage multiple drafts or versions of a compound document and must provide for the election and creation of a compound document draft together with management and reconciliation of multiple drafts. These are closely related to document storage interfaces, which manage the storage and retrieval of a compound document and its parts. Document storage interfaces must be abstracted from the underlying storage systems sufficiently to support a multiplicity of different storage systems. In the OpenDoc reference implementation, the storage system is constructed using Bento containers, for instance.

Detailed Description Text (14):
OpenDoc.TM. Compound Document Part Interfaces

Detailed Description Text (16):
The independent part developer starts with "Part" 36 to create a new OpenDoc part. Class type "Part" 36 has sixty-two methods as part of its interface and, for simple new parts and quick prototypes, only a dozen or so of these interfaces need be changed to create a new part. A new part handler must draw its part, receive its part's data, activate its part and handle events sent to its part. It also needs some sort of command or user interface and must provide menus for the menu bar, optional windows, dialogues and palettes, and must provide for containing other parts as necessary. The part editor must embed frames, create facets for visible frames and store frames for embedded parts if any. Beyond these capabilities, the new part handler may support asynchronous drawing, drag-drop and cut-paste operations, scripting and linkages to other parts. An existing application may be simply converted to an OpenDoc part handler by starting with one of the class types in FIG. 1 and "subclassing it." This ensures correct default behavior for content-independent actions and allows for rapidly adding application functionality.

Detailed Description Text (17):
FIG. 2 shows the message transfers between various class types at runtime in the OpenDoc architecture. For instance, the Part object 40 accepts messages from no more than one LinkSource object 42, no more than one Link object 44 and no more than one Frame object 46. Part object 40, together with any other embedded Part objects (not shown), transmits messages to the SemanticInterface object 48, the StorageUnit object 50 and Frame object 46. The messages to Frame object 46 include messages 52 for its own Frame and messages 54 for the frames of all embedded parts within part 40.

Detailed Description Text (18):
FIG. 3 shows the Bento object container 56 known in the art. Bento container 56 can be thought of as an "envelope" for objects and it may be generally applied to any object-oriented programming system, including the OpenDoc document-centric programming system. Bento container 56 includes standard procedures for moving itself, for viewing the inventory list, for removing things from and adding new things into the envelope. All such procedures are independent of container contents. In a document-centric architecture, Bento container 56 serves three functions. It is the reference file storage system, the object container for things placed on the clipboard and the container of choice for transporting compound documents across platforms.

Detailed Description Text (19):
The Bento methods may be considered as a file system within a file but it is actually a flat structure and not a tree structure reflecting the embedding relationship. In FIG. 3, Bento container 56 includes a plurality 58 of objects, exemplified by Objects A and B. Each Object has one or more properties associated with that object and each such property has a set of values or data. For instance,

Object A includ[redacted] Properties p, q, and r. Proper[redacted] q may be formatted text stored in several different formats; e.g., RTF format 60, SGML format 62, and ASCII format 64, which could include a pointer 66 to another Bento container (not shown). In the OpenDoc system, StorageUnit object 50 (FIG. 2) is similar to Bento Container 56 (FIG. 3) but is also adapted to function as the front end for other storage types such as in an object-oriented database management system or in another object storage system such as the OLE DocFile Format.

Detailed Description Text (20):
FIG. 4 shows the structure of the OpenDoc StorageUnit class type embodied as object 50. Object 50 includes two "Property" objects 70 and 72. Property object 72 includes two embedded Value objects 74 and 76. Value object 76 is linked to another StorageUnit object 78, which is itself embedded in the same current draft object 80 containing StorageUnit 50. StorageUnit object 78 is also linked to two other commonly-embedded StorageUnits 82 and 84.

Detailed Description Text (21):
FIG. 5 shows a "canonical" compound document display 86 according to the OpenDoc model, which enables the creation of compound documents that are created and edited by several cooperating applications working within a single compound document.

Detailed Description Text (23):
The key to the notion of parts is that each part of a compound document has its own "content model," which is the model of objects and operations that is presented to a user of the part. The content model changes at the boundary between parts. For instance, in document 86, text part 98 includes lines, words, paragraphs, characters, and embedded button part 100. Internally, the part handler in charge of part 98 may have a model that includes run-length encoded arrays for style information, line end arrays and similar tools. These are not presented to the user so they are not part of the content model. In the root graphics part of the compound document, the content objects may look very different. Circles, rectangles, and lines are content objects that could be provided in such a graphics part.

Detailed Description Text (26):
A part and its handler together form the equivalent of a programmatic object (data and methods) in the object-oriented programming sense. The part provides the state information while the part-handler provides the methods or behavior. When bound together, they form an editable segment of a compound document. Part handlers are dynamically linked into the runtime world of the compound document based on the part types that appear in the document. Because any type of part may appear in any document at any time, the part handlers must be dynamically linked to provide a smooth user interface. The OpenDoc system assumes that parts are mainly used in a shell that allows a compound document to act like a single application. This document shell provides an address base, distributes events and provides basic user interface resources such as windows and menus.

Detailed Description Text (27):
Because one of the important features of the OpenDoc system is customized documents, the part handlers must handle two distinct kinds of events: semantic events and user-interface (UI) events. The difference between the two event types relates to whether the event makes sense outside of the particular document display context. A UI event requires information about where windows are located in the display, how parts of the document have been scrolled, or in what part a selection was last made. These events include mouse clicks, keystrokes and menu activations. In contrast, a semantic event relates to the semantics of the compound document and is generally independent of the graphical context. Semantic events relate to the content model of the part.

Detailed Description Text (30):
Given the presence of multi-part compound documents, a persistent storage mechanism is needed to enable multiple part handlers to share a single compound document file. OpenDoc assumes that such a storage system can effectively give each part its own storage stream and that reliable references can be made from one such stream to another. Because many code sections may need access to a given part, the storage system must support a robust annotation scheme allowing information to be associated with a part without disturbing the part format.

Detailed Description Text (31):
At runtime, OpenDoc assumes that an instance of the document shell is created for

each document. This generic shell must provide four basic structures to the part handlers: (a) the storage system, (b) the window and its associated state, (c) the event dispatcher and (d) an arbitration registry for negotiating shared resources such as the menu bar. The runtime shell may also be responsible for binding and loading part handlers for the parts that appear in the compound document. It is assumed that once a given part handler is loaded, any part in any document may share the executable code of the part handler.

Detailed Description Text (36):
Because OpenDoc separates the part handlers from document level functions, new features can be added to compound documents without revising of existing applications. OpenDoc's document shell thus provides access to existing applications without modification. OpenDoc also allows for collaboration between parts through scripting. Scripting forms a rich medium for coordinating the work of parts in documents and allows users and parts to work together to perform tasks.

Detailed Description Text (37):
The Open Scripting Architecture (OSA) provides an open architecture for scripting languages such as OREXX, Apple Script, OLE Automation, and Scriptx. The OSA includes two components. The first is the typing of script types that permits inclusion of the correct scripting engine for script processing. The second is the standardization of script semantic messages in the Open Event form. Scripts become part of a compound document, either attached or embedded as necessary. They may be used to implement control structures or used as instructions for embedding into documents that are mailed or transferred out of the network.

Detailed Description Text (38):
The Open Events model is based on the standard registry of verbs and object classes. Open Events are arranged in application suites that include such application groups as "Required," "Core," "Text," "Table," "Database," "Compound Document," and many others yet to be invented. Each event includes three elements. These are (a) event verbs such as "Open," "Close," "Select," "Get," "Put," and the like, (b) object classes or object specifiers such as "Application," "Document," "Word," "Paragraph," "Circle," and the like, and (c) descriptor types such as Boolean Fixed Attribute "greater than," "3.sup.rd," "bold," and the like.

Detailed Description Text (45):
A new compound document part, herein denominated the CHRON part, allows the user working on a compound document to specify "script events" with which the operating system automatically executes designated scripts at predetermined times. FIG. 7 shows a compound document object 120 that includes two CHRON objects 122 and 124. Document 120 also by way of example is shown having an embedded text part 126, an embedded video part 128, a table part 130, and an image part 132. Each of the parts within document 120 may be opened by the user to view or edit its contents. The document-centric system automatically links each object to the part handler necessary for editing and/or viewing the content of the object. For instance, text object 126 can be opened by the user with a mouse click. The system then may respond to the mouse click by launching a part handler to display the text content of object 126 and/or launching a text editing application program to accept keyboard and mouse editing commands from the user. Both display and editing can be limited to a predefined window in the user display.

Detailed Description Text (48):
The CHRON class type also includes one or more scheduled-time parts (time/date), as exemplified by scheduled-time parts 134 and 136, each of which specifies when a script is to be executed. CHRON part 122 may also include a table part (not shown) that specifies the relationship between the two scheduled-time parts 134 and 136 and the single script part 138, for example. An instance of the CHRON class type may be inserted into any container part within a compound document, including the root document itself exemplified by document 120. Thus, although not shown, if script event 142 in CHRON part 124 causes the execution of a word search in the content model of text part 126, then CHRON part 124 could be embedded within text part 126 itself rather than in root document 120.

Detailed Description Text (53):
Another important feature of the CHRON class type of this invention is its reusability. That is, each instance may be edited, removed and inserted as directed by the user. New instances may be spanned as necessary and edited by the user. FIGS. 9A-9D provide a rudimentary illustration of the user interface suitable for editing

either of the several objects embedded in CHRON class type of this invention. For instance, in FIG. 9A, the user points to the scheduled-time window 150 and clicks the mouse to launch the scheduled-time object editor program (not shown). This editing program creates a user interface window, which is exemplified by window 152 in FIG. 9C. Window 152 includes the data field 154, the time field 156 and a register button 158. Clicking on either field 154 or 156 launches a field editor that accepts keyboard input in the usual manner. Clicking on register button 158 updates the persistent events stored by the system. This procedure is discussed below in connection with FIG. 10.

Detailed Description Text (56):
If the cursor is neither on field 150 or field 160, then step 176 tests to see if the user is pointing to register field 158 (FIG. 9C), either within window 152 or elsewhere if possible. If so, then step 178 checks the temporary time/date objects created in step 174 to ensure their validity with respect to some syntax. Step 180 steers the process to step 182, generating an error indication and exiting at step 170 if the temporary time/date objects are not valid. If valid, then step 184 tests a "persistent state" (PS) flag, which is set only when the CHRON object being edited is linked to existing persistent events in system storage. The PS flag is a feature that permits each CHRON part to be reused repeatedly.

Detailed Description Text (58):
While the invention discussed above is primarily disclosed as a process, practitioners of ordinary skill in the art can appreciate that an apparatus or system, such as the system illustrated in FIG. 11, can be programmed or otherwise designed to facilitate the practice of the process of this invention. For instance, system 200 in FIG. 11 may include a central processing Unit (CPU) 202 linked to a graphical user interface (GUI) display 204 and a memory system 206. The system user may view and edit compound documents from a keyboard 208 and a mouse 210 in combination with GUI display 204. Memory 206 may be linked to a persistent data storage subsystem 212 in the usual manner. In a compound document system, memory 206 may include numerous program objects, exemplified by a storage manager program object 214 for controlling data transfers between memory 206 and persistent memory 212, an operating system object 216 for controlling all system activities, and a plurality of class type instances, otherwise herein denominated "objects," exemplified by object 218. By way of example, object 218 may represent a compound document including an instance of the CHRON class type of this invention. It can be understood and appreciated that a computer system exemplified by system 200 in FIG. 11 also falls within the spirit and scope of this invention.

CLAIMS:

1. In a machine-implemented document-centric application processing system including a data processor coupled to a user display, to means for accepting user requests and to memory means including persistent storage means for storing an object-oriented operating system and a class library of objects including a plurality of data objects and a plurality of program objects, said memory means including a script-editing program object, a schedule-time object editor, an icon-display program object, an event-scheduling system and a plurality of parts and part editors, a machine-executed method for scheduling script execution in a compound document object, said method comprising the steps of:

(a) spawning in said memory means an instance of a reusable CHRON object including at least one script object and at least one schedule-time object, said instance being contained in said compound document object;

(b) creating for display on said user display a CHRON icon having at least one schedule field for displaying at least part of said at least one schedule-time object;

(c) responsive to a user request to revise said at least one schedule-time object, performing the steps of

(c.1) spawning in said memory means an instance of said schedule-time object editor and

(c.2) updating a temporary schedule-time object in said memory means according to at least one corresponding said user request;

(d) comparing said temporary schedule-time object to a valid schedule-time object format rule;

(e) returning an error condition to said user display if said temporary schedule-time object form is invalid; otherwise

(f) sending a message to said event-scheduling system requesting the creation of a persistent event object for executing said at least one script object according to the schedule specified by said temporary schedule-time object.

5. A reusable compound document script scheduling object for scheduling script execution in a compound document object in a machine-implemented document-centric application processing system including a data processor coupled to a user display, to means for accepting user requests and to memory means including persistent storage means for storing an object-oriented operating system and a class library of objects including a plurality of data objects and a plurality of program objects, said memory means including a script-editing program object, a schedule-time object editor, an icon-display program object, an event-scheduling system and a plurality of parts and part editors, said reusable script scheduling object comprising:

at least one script object having information for specifying a script to be executed at a predetermined time;

at least one schedule-time object having information for specifying at least one predetermined time at which the associated said script object is scheduled for execution;

object container means stored in said memory means for containing said at least one script object and said at least one schedule-time object and for linking the components of said reusable script scheduling object with other said parts contained in said compound document; and

registration means coupled to said event-scheduling system for creating a persistent event object in said persistent storage means for executing said script at said predetermined time.

6. The reusable script scheduling object of claim 5 further comprising:

icon display means for creating for display on said user display a CHRON icon having at least one schedule field for displaying at least part of said at least one schedule-time object; and

GUI linkage means coupled to said schedule-time object for launching said schedule-time object editor for editing said schedule-time object responsive to a user request.

8. The reusable script scheduling object of claim 7 wherein said GUI linkage means includes means for launching said script-editing program object to edit said script object responsive to a user request.

9. The reusable script scheduling object of claim 6 wherein said GUI linkage means includes means for launching said script-editing program object to edit said script object responsive to a user request.

10. An object-oriented compound document processing system comprising:

a data processor;

a user display coupled to said data processor;

means for accepting user requests coupled to said data processor;

memory means including persistent storage means coupled to said data processor for storing an object-oriented operating system and a class library of objects including a plurality of data objects and a plurality of program objects;

an event-scheduling system coupled to said object oriented operating system in said memory means for creating persistent event objects in said persistent storage means;

a plurality of compound documents each containing one or more parts in said memory means;

a plurality of part editors each for editing a particular part type in said memory means;

one or more reusable script-scheduling objects each for scheduling script execution in a compound document;

at least one script object in each said reusable script-scheduling object for specifying a script to be executed at a predetermined time;

at least one script-editing program object in said memory means for editing said script objects in said memory means;

at least one schedule-time object in each said reusable script scheduling-object for specifying at least one predetermined time at which an associated said script object is scheduled for execution;

at least one schedule-time object editor in said memory means for editing said schedule-time objects; and

registration means in each said reusable script scheduling object coupled to said event-scheduling system for creating a persistent event object in said persistent storage means for executing said script at said predetermined time.

11. The object-oriented compound document processing system of claim 10 further comprising:

icon display means for creating for display on said user display a CHRON icon having at least one schedule field for displaying at least part of said at least one schedule-time object;

GUI linkage means coupled to said schedule-time object for launching said schedule-time object editor for editing said schedule-time object responsive to a user request.

12. The object-oriented compound document processing system of claim 11 wherein each said CHRON icon schedule field comprises:

a time-of-day display when said schedule-time object schedules the daily execution of the corresponding said script object at said time-of-day;

a time-date-year display when said schedule-time object schedules a single execution of the corresponding said script object at said time-date-year;

a date display when said schedule-time object schedules the annual execution of the corresponding said script object at said date;

a weekday display when said schedule-time object schedules the weekly execution of the corresponding said script object; and

no time display when said schedule-time object schedules execution of the corresponding said script object according to an internal tabular specification.

13. The object-oriented compound document processing system of claim 12 wherein said GUJI linkage means includes means for launching said script-editing program object to edit said script object responsive to a user request.

14. The object-oriented compound document processing system of claim 11 wherein said GUI linkage means includes means for launching said script-editing program object to edit said script object responsive to a user request.

15. A computer program product for use in scheduling script execution in compound document objects in a machine-implemented document-centric application processing system including a data processor coupled to a user display, to means for accepting user requests and to memory means including persistent storage means for storing an object-oriented operating system and a class library of objects including a

plurality of d[redacted] objects and a plurality of pr[redacted]am objects, said memory means including a script-editing program object, a schedule-time object editor, an icon-display program object, an event-scheduling system and a plurality of parts and part editors, said computer program product comprising:

a recording medium;

means recorded on said recording medium for directing said document-centric application processing system to spawn in said memory means an instance of a reusable CHRON object including at least one script object and at least one schedule-time object, said instance being contained in said compound document object;

means recorded on said recording medium for directing said document-centric application processing system to create for display on said user display a CHRON icon having at least one schedule field for displaying at least part of said at least one schedule-time object;

means recorded on said recording medium for directing said document-centric application processing system to perform the steps of, responsive to a user request to revise said at least one schedule-time object,

spawning in said memory means an instance of said schedule-time object editor and

updating a temporary schedule-time object in said memory means according to at least one corresponding said user request;

means recorded on said recording medium for directing said document-centric application processing system to compare said temporary schedule-time object to a valid schedule-time object format rule in said memory means;

means recorded on said recording medium for directing said document-centric application processing system to perform the steps of

returning an error condition to said user display if said temporary schedule-time object form is invalid, otherwise

sending a message to said event-scheduling system requesting the creation of a persistent event object for executing said at least one script object according to the schedule specified by said temporary schedule-time object.

Generate Collection | Print

L5: Entry 5 of 10 File: USPT Feb 10, 1998

US-PAT-NO: 5717755
DOCUMENT-IDENTIFIER: US 5717755 A

TITLE: Distributed cryptographic object method

DATE-ISSUED: February 10, 1998

INVENTOR-INFORMATION:

NAME	CITY	STATE	ZIP CODE	COUNTRY
Shanton; M. Greg	Fairfax	VA		

ASSIGNEE-INFORMATION:

NAME	CITY	STATE	ZIP CODE	COUNTRY	TYPE CODE
TECSEC, Inc.	Vienna	VA			02

APPL-NO: 08/ 304867 [PALM]
DATE FILED: September 13, 1994

PARENT-CASE:
This is a continuation of U.S. patent application Ser. No. 08/138,857 filed Oct. 18, 1993; now U.S. Pat. No. 5,363,702.

INT-CL: [06] H04 L 9/32, H04 L 9/00

US-CL-ISSUED: 380/25; 380/4, 380/9, 380/21, 380/23, 380/28, 380/49, 380/50, 340/825.31, 340/825.34
US-CL-CURRENT: 713/166; 340/5.26, 380/277, 380/28

FIELD-OF-SEARCH: 380/3, 380/4, 380/5, 380/9, 380/10, 380/21, 380/23, 380/24, 380/25, 380/28, 380/30, 380/43, 380/49, 380/50, 340/825.31, 340/825.34, 455/33.3

PRIOR-ART-DISCLOSED:

U.S. PATENT DOCUMENTS

Search Selected | Search ALL

	PAT-NO	ISSUE-DATE	PATENTEE NAME	US-CL
☐	4218582	August 1980	Hellman et al.	380/30
☐	4405829	September 1983	Rivest et al.	380/30
☐	4424414	January 1984	Hellman et al.	380/30
☐	4713753	December 1987	Boebert et al.	380/4 X
☐	4864616	September 1989	Pond et al.	380/25
☐	4955082	September 1990	Hattori et al.	455/33.3
☐	4962533	October 1990	Krueger et al.	380/25
☐	4984272	January 1991	McIlroy et al.	380/25
☐	5052040	September 1991	Preston et al.	380/4
☐	5065429	November 1991	Lang	380/25
☐	5191611	March 1993	Lang	380/25
☐	5204961	April 1993	Barlow	380/4 X
☐	5369702	November 1994	Shanton	380/4
☐	5369707	November 1994	Follendore, III	380/25

ART-UNIT: 222

PRIMARY-EXAMINER: Gregory; Bernarr E.

ABSTRACT:

A system for increasing the security of a computer system, while giving an individual user a large amount of flexibility and power. To give users the most power and flexibility, a standard object that has the capability to embed objects is used. To allow users even more flexibility, a standard object tracking mechanism is used that allows users to distribute multiple encrypted embedded objects to other individuals in a single encrypted object. By effecting compartmentalization of every object by label attributes and algorithm attributes, multi-level multimedia security is achieved.

20 Claims, 8 Drawing figures

L5: Entry 5 of 10 File: USPT Feb 10, 1998

DOCUMENT-IDENTIFIER: US 5717755 A
TITLE: Distributed cryptographic object method

DATE FILED (1):
19940913

Brief Summary Text (6):
Cryptographic systems have evolved to fill the needs of companies and individuals wanting to protect the proprietary commercial information of a company from competitors and those who generally should not have that information. Encryption of data is therefore a critical requirement in denying access to confidential information from those who are not so authorized. Cryptographic "keys" are an essential part of the information encryption process. The cryptographic key, or "key" for short, is a sequence of letters, numbers, or bytes of information which are manipulated by a cryptographic algorithm to transform data from plain (readable) text to a series of unintelligible text or signals known as encrypted or cipher text. The key is then used by the receiver of the cipher text to decrypt the message back to plain text. However, for two people to communicate successfully using keys, each must use the same key, assuming that the same encryption/decryption algorithm is used on both ends of the communication.

Brief Summary Text (26):
In object linking and embedding, an object can be any userselected group of data, such as a block of text, a set of spreadsheet cells, a chart, sounds, or a graphical image. This data can be embedded in or linked to another document created by a different application. For example, a folder may represent a directory and may contain a group of files, or it may represent a group of programs. Folders can also contain other folders.

Detailed Description Text (32):
Microsoft's Object Linking and Embedding 1.0(OLE 1.0)

Detailed Description Text (33):
Microsoft's Object Linking and Embedding 2.0(OLE 2.0)

Detailed Description Text (38):
Macintosh's Compound Document Standard

CLAIMS:

1. A method for providing multi-level multimedia security in a data network, comprising:

A) accessing an object-oriented key manager;

B) selecting a first object to encrypt;

C) selecting a first label for the first object;

D) selecting an encryption algorithm;

E) encrypting the first object according to the encryption algorithm;

F) labelling the encrypted first object wherein the labelling comprises creating a display header;

G) reading the first object label;

H) determining access authorization based on the first object label; and

I) allowing access to the first object only if access authorization is granted.

4. The method of claim 3, further comprising

A) reading the second object label;

B) determining access authorization based on the second object label; and

C) allowing access to the second object only if access authorization is granted.

9. The method of claim 8, further comprising:

A) selecting a second label for the second object;

B) selecting an encryption algorithm;

C) encrypting the second object;

D) labelling the second encrypted object with a second object label;

E) reading the second object label;

F) determining access authorization based on the second object label; and

G) uncovering the second object only if access authorization is granted.

14. A system for providing multi-level multimedia security in a data network, comprising:

A) a data processor, the data processor comprising:

1)a system memory, comprising stored data;

2)an encryption algorithm module, comprising logic for converting unencrypted objects into encrypted objects, the encryption algorithm module being disposed to access data stored in the system memory;

3)an object labelling subsystem, comprising logic means for limiting object access, subject to label conditions, the object labelling subsystem being disposed to access data stored in the system memory and the object labelling subsystem being further disposed to accept inputs from the encryption algorithm module;

4)a decryption algorithm module, comprising logic for converting encrypted objects into unencrypted objects, the decryption algorithm module being disposed to access data stored in the system memory means; and

5)an object label identification subsystem, comprising logic for limiting object access, subject to label conditions, the object label identification subsystem being disposed to access data stored in the system memory and the object label identification subsystem being further disposed to accept inputs from the decryption algorithm module;

B) the encryption algorithm module working in conjunction with the object labelling subsystem to create an encrypted object such that the object label identification subsystem limits access to an encrypted object.

18. A system for providing multi-level multimedia security in a data network, comprising:

A) means for accessing an object-oriented key manager;

B) means for selecting a first object to encrypt;

C) means for selecting a label for the first object;

Not OLE

D) means for selecting an encryption algorithm;

E) means for encrypting the first object;

F) means for embedding the first object within a second object;

G) means for labelling the encrypted first object;

H) means for reading the label;

I) means for determining access authorization based on the label; and

J) means for accessing the first object only if access authorization is granted;

K) the means for embedding the first object within a second object including means for covering the first object with a second object.

19. The system of claim 18, further comprising:

A) means for selecting the second object to encrypt;

B) means for selecting a second label for the second object;

C) means for encrypting the second object;

D) means for labelling the encrypted second object;

E) means for reading the second label;

F) means for determining access authorization based on the second label; and

G) means for accessing the second object only if access authorization is granted.

Cited

Generate Collection | Print

L5: Entry 6 of 10 File: USPT Nov 11, 1997

US-PAT-NO: 5687331
DOCUMENT-IDENTIFIER: US 5687331 A

TITLE: Method and system for displaying an animated focus item

DATE-ISSUED: November 11, 1997

INVENTOR-INFORMATION:

NAME	CITY	STATE	ZIP CODE	COUNTRY
Volk; Patrick M.	Kirkland	WA		
Robin; Michael Breed	Redmond	WA		
Thorne, III; Edwin	Seattle	WA		
Kapell; JoGene	Bellevue	WA		

ASSIGNEE-INFORMATION:

NAME	CITY	STATE	ZIP CODE	COUNTRY	TYPE CODE
Microsoft Corporation	Redmond	WA			02

APPL-NO: 08/ 510965 [PALM]
DATE FILED: August 3, 1995

INT-CL: [06] G06 F 3/00

US-CL-ISSUED: 395/327; 395/977, 395/962
US-CL-CURRENT: 345/840; 345/823, 345/861, 345/962, 345/977

FIELD-OF-SEARCH: 395/159, 395/333, 395/334, 395/339, 395/349, 395/327, 395/977, 395/962

PRIOR-ART-DISCLOSED:

U.S. PATENT DOCUMENTS

Search Selected | Search ALL

	PAT-NO	ISSUE-DATE	PATENTEE-NAME	US-CL
☐	5479602	December 1995	Baecker et al.	395/159
☐	5491795	February 1996	Beaudet et al.	395/159
☐	5524195	June 1996	Clanton, III et al.	395/159 X
☐	5555496	September 1996	Tackbary et al.	395/227
☐	5564004	October 1996	Grossman et al.	395/159

ART-UNIT: 245

PRIMARY-EXAMINER: Bayerl; Raymond J.

ASSISTANT-EXAMINER: Katbab; A.

ABSTRACT:

A viewer interface is disclosed for use in an interactive television network operative for providing an animated focus item in association with a control item to indicate that the control item is in a state responsive to commands from a user input device. An "animation" is any form of highlighting that is non-static, including but not limited to flashing, varying illumination, varying size, varying shape, varying position, varying color, varying display components, a moving and/or changing cartoon type image, a video image, a sound track, or a combination of these elements. Selection of the control item to receive focus and selection of options presented by control items having focus are accomplished by viewer interaction with the remote control unit, and such selections do not require a keyboard or mouse to indicate the viewer's desire to change the focus from one control item to another or to select an option. The user interface is also suitable for use in a general computing environment as well as in an interactive television environment.

77 Claims, 30 Drawing figures

L5: Entry 6 of 10 File: USPT Nov 11, 1997

DOCUMENT-IDENTIFIER: US 5687331 A
TITLE: Method and system for displaying an animated focus item

DATE FILED (1):
19950803

Brief Summary Text (29):
Third, programmers of applications running on traditional computer systems do not have a particularly strong motivation to attract a user's attention to a particular control item. In comparison, focus items in an interactive television environment may be called on to serve a promotional or advertisement related function. Similarly, application programmers of interactive television programs may want to customize the viewer interface in accordance with the theme of the underlying program. For example, a music program may use an animated musical instrument as a focus item, and a sports program may use an animated athlete as a focus item. Therefore, the method of indicating the focus in an interactive television environment will be improved over traditional computer graphical user interface focus methods if the focus items can be easily animated in accordance with the theme of the underlying program with a promotional motivation. In particular, it would be advantageous if sponsors or authors of individual application programs could customize the focus item that is displayed during a particular application or portion thereof.

Detailed Description Text (27):
Turning now to the drawings, in which like numerals indicate like elements throughout the several FIGS., FIG. 1 illustrates the operating environment for an interactive television network, a preferred embodiment of the present invention. Referring to FIG. 1, the interactive television network 10 includes a headend system 12 for delivering programming information to and receiving instructions from a consumer system 14 via a "two-way" distribution network 16. The headend system 12 is the control center for collecting, organizing, and distributing the signals for all interactive network operations and the source for all programming information. The distribution network 16 transports signals carrying programming information and instructions between the headend system 12 and the consumer system 14. The distribution network 16 may include a world-wide public asynchronous transfer mode (ATM) compatible network with links to the Internet, third party service providers, and other wired and wireless communications networks. The consumer system 14 includes the equipment required for a consumer to receive programming information directly at his or her office or residence and to transmit requests and instructions to the headend system 12.

Detailed Description Text (28):
The headend system 12 can include a set of headend servers 20, including a continuous media server (CMS) system 22, and one or more administrative servers 24 to support various network functions, and a control network 26 linking these headend servers. The headend servers 20 can execute program modules, including service and application program software, to support the transmission of programming information and the reception of requests for such programming information.

Detailed Description Text (29):
It will be appreciated that the headend servers 20 are not necessarily located in one physical location but can be linked by wired and/or wireless communications paths supplied by the control network. The control network 26 can be a local area network, a wide area network, or a combination of both types of networks. For the preferred embodiment, the control network 26 is implemented as an ATM-based network

for routing digital data between the headend servers 20 and the distribution network 16.

Detailed Description Text (32):
To support the tasks of updating or revising programming information stored on a memory storage device 30 of the CMS system 22, a computer workstation 32 and a remote server 34 can be connected to the control network 26 via a communications link 36. This communications link allows a program distributor or supplier, which typically operates at a location remote from the CMS system 22, to transmit programming information for storage by one or more of the memory storage devices 30 and for eventual distribution to consumers via the headend system 12. The communications link 36 can be implemented by either a wireless or wired communications system. For example, the communications link 36 can be constructed as a microwave link, a leased line link, a fiber optic link, a wire link, or as a conventional telephone link.

Detailed Description Text (33):
The administrative servers 24 of the headend system 12 can support a variety of services and applications associated with the interactive television network 10, including network security, monitoring, object storage, financial transactions, data management, and other administrative functions. The administrative servers 24 also handle the interactive service requests or instructions transmitted via the consumer system 14 by consumers. For an application involving a large base of consumers, an administrative server 24 is preferably dedicated to a particular service or function. For example, one or more servers can handle all consumer authorization requirements, and other servers can handle network management services and so forth. These administrative servers preferably support the Simple Network Management Protocol (SNMP) to enable end-to-end network administration and monitoring.

Detailed Description Text (47):
The set-top terminal 48 can be connected to a peripheral device via input/output (I/O) ports 74. The I/O ports 74 support the connection of the system bus 70 to a connected peripheral device. For example, the output device 50 can be connected to the I/O ports 74 via a conductor 52. Likewise, an input device 54, such as a game control 90, can be connected to the I/O ports 74. In contrast to the remote control 80, which communicates with the remote control receiver 60 via a wireless communications link, other types of input devices 54 are typically connected to the I/O ports 74 via a cable. Nevertheless, those skilled in the art will appreciate that input devices 54 can communicate with the set-top terminal 48 by use of either wireless or wired communications links.

Detailed Description Text (76):
Referring to FIG. 2, the set-top terminal 48 supports a graphical viewer interface 100 for the interactive television system 10. Graphical user interfaces for conventional computer systems are well known in the desktop computer environment. There are a number of differences between a conventional graphical user interface and an interactive television graphical viewer interface according to the preferred embodiment of the present invention. For example, the interactive television graphical viewer interface does not include the following traditional elements of a desktop computer base graphical user interface: system menus, menu bars, multiple document interface, iconized windows, system user interface for changing window size or position, dynamic data exchange, advanced object linking and embedding features (e.g., compound documents), a clipboard, or double-click input capabilities.

Not analog art?

Detailed Description Text (122):
Thus, frames are building blocks of the preferred interface. Frames can contain graphic images, animations, video, text, and other standard elements as well as other frames. Control and focus objects are typically constructed by starting with a frame as a parent container and then adding additional elements to the frame. Frames can be positioned in absolute screen coordinates or relative to their parent. For example, a focus frame may be linked to a control frame so that the focus item is always displayed in the same manner relative to the control item. Thus, if the control item is moved, the linked focus item will move with it.

Detailed Description Text (135):
Referring again to FIG. 7, the control manager 216 handles the linking and layering of control and focus items in the system. A focus object 206 may be linked to a control object, which becomes a parent container holding the focus object. An object known as TvTop is provided as a master container maintained by the control manager.

Top-level frame[redacted]i.e., frames with no parent) a[redacted]registered with TvTop so that the control manager can facilitate navigating and switching between applications when more than one application is running at a time.

Detailed Description Text (173):
After completing the registration process and presenting an initial display of the graphical viewer interface 100, the set-top terminal 48 is now ready to accept a command from the input device 54. In step 1210, the system receives a command signal in response to the viewer's decision to conduct a desired control operation. Note that the steps shown on FIG. 9B, including the steps from 1210 to "END", are "distributed" in that they are executed only in the context of a control object having the focus. The control object having the focus can call upon system services 1300 and 1400 to assist in processing input and managing focus selections.

Detailed Description Text (200):
FIG. 14 is a logical flow diagram that illustrates the steps of computer-implemented routine 1430 for displaying a focus item associated with a control item holding focus. There are two ways for a custom focus item to be specified by the system: (1) on registering with the focus manager, a control object may specify a focus object to be used when that control object has the focus (note that the focus object may be passed to the control object by another program module prior to the control object registering with the focus manager), or (2) an application program module may monitor focus change events and render the appropriate focus item directly. The former approach is the preferred embodiment whereby the control object 210 contains information related to the display of the focus item. Therefore, in the preferred embodiment this information is retrieved at step 1423 (FIG. 13), and routine 1430 has all of the information it needs to animate the focus item. Accordingly, routine 1430 begins with routine 1431 wherein the focus item may be animated. The focus object and the control object having the focus are then linked in step 1438, and the routine returns.

Generate Collection | Print

L5: Entry 7 of 10 File: USPT Oct 21, 1997

US-PAT-NO: 5680452
DOCUMENT-IDENTIFIER: US 5680452 A

TITLE: Distributed cryptographic object method

DATE-ISSUED: October 21, 1997

INVENTOR-INFORMATION:

NAME	CITY	STATE	ZIP CODE	COUNTRY
Shanton; M. Greg	Manassas	VA		

ASSIGNEE-INFORMATION:

NAME	CITY	STATE	ZIP CODE	COUNTRY	TYPE CODE
TECSEC Inc.	Vienna	VA			02

APPL-NO: 08/ 394402 [PALM]
DATE FILED: February 24, 1995

PARENT-CASE:
This is a continuation-in-part of U.S. patent application Ser. No. 08/304,867, filed Sep. 13, 1994, which is a continuation of U.S. patent application Ser. No. 08/138,857, filed Oct. 18, 1993, which issued as U.S. Pat. No. 5,369,702 on Nov. 29, 1994.

INT-CL: [06] H04 L 9/00

US-CL-ISSUED: 380/4; 380/9, 380/21, 380/23, 380/25, 380/28, 380/49, 380/50, 340/825.31, 340/825.34
US-CL-CURRENT: 713/167; 340/5.74, 380/269, 380/28

FIELD-OF-SEARCH: 380/3, 380/4, 380/5, 380/9, 380/10, 380/23, 380/24, 380/25, 380/21, 380/49, 380/50, 380/28, 380/30, 380/43, 340/825.31, 340/825.34, 455/33, 455/3

PRIOR-ART-DISCLOSED:

U.S. PATENT DOCUMENTS

Search Selected | Search ALL

	PAT-NO	ISSUE-DATE	PATENT NAME	US-CL
☐	4218582	August 1980	Hellman et al.	380/30
☐	4405829	September 1983	Rivest et al.	380/30
☐	4424414	January 1984	Hellman et al.	380/30
☐	4713753	December 1987	Boebert et al.	380/4 X
☐	4864616	September 1989	Pond et al.	380/25
☐	4955082	September 1990	Hatoori et al.	455/33.3
☐	4962533	October 1990	Krueger et al.	380/25
☐	4984272	January 1991	McIlroy et al.	380/25
☐	5052040	September 1991	Preston et al.	380/4
☐	5065429	November 1991	Lang	380/25
☐	5191611	March 1993	Lang	380/25
☐	5204961	April 1993	Barlow	380/4 X
☐	5369702	November 1994	Shanton	380/4
☐	5369707	November 1994	Follendore III	380/25

ART-UNIT: 222

PRIMARY-EXAMINER: Gregory; Bernarr E.

ABSTRACT:

A system for increasing the security of a computer system, while giving an individual user a large amount of flexibility and power. To give users the most power and flexibility, a standard object that has the capability to embed objects is used. To allow users even more flexibility, a standard object tracking mechanism is used that allows users to distribute to other individuals multiple encrypted objects embedded in a single encrypted object. By effecting compartmentalization of every object by label attributes and algorithm attributes, multi-level multimedia security is achieved. Label attributes are used to restrict access to objects based on location, group, or other criteria and may specify personal access. Access type, such as read-only, write-only, and print-only may be specified. Nested embedded objects may be accessed directly through selection from a header array.

17 Claims, 8 Drawing figures

L5: Entry 7 of 10 File: USPT Oct 21, 1997

DOCUMENT-IDENTIFIER: US 5680452 A
TITLE: Distributed cryptographic object method

DATE FILED (1):
19950224

Brief Summary Text (6):
Cryptographic systems have evolved to fill the needs of companies and individuals wanting to protect proprietary commercial information from competitors and others who generally should not be privy to that information. Such systems ensure that data is not recognizable to unauthorized parties who intercept it. Only authorized persons would have the means for decrypting the data and returning it to readable form.

Brief Summary Text (8):
Encryption of data is therefore a critical requirement in controlling access to confidential information. Cryptographic "keys" are an essential part of the data encryption process. The cryptographic key, or "key" for short, is a data stream that is manipulated with plain (readable) text data according to a cryptographic algorithm to transform the plain text data to a string of unintelligible text or signals known as encrypted text or cipher text. The key is then used by an authorized receiver of the cipher text to decrypt the message, returning it to plain text form. However, for two people to communicate successfully using such a cryptographic system, each must use the same key, assuming that the same encryption/decryption algorithm is used on both ends of the communication. Thus, a system must be put into place for the distribution and other management of these keys within the closed group of authorized users.

Brief Summary Text (23):
The present invention is a data processing system which can be used to select and encrypt objects. The encrypted objects, either individually or in groups, can be embedded in other objects called container objects, which may also be encrypted. This process is based on a technique called cryptographic encapsulation. The original representation of the object embedded within this container object can then be deleted, as all of the contained object information can be found in encrypted form within the container object. The container object can be further embedded within another container object. Container objects can only be "opened", that is, decrypted, by users having access authority in the form of a cryptographic key and who are using the proper cryptographic engine. Existence of the embedded object is not even known to users who don't have access to the container. The present invention can be utilized with all objects in the system, including standard software applications, utilities, operating systems, and devices. Device objects can be embedded within other device objects, or within a data file object. For example, a printer object may be embedded within a file object; only a user having cryptographic access to the data file will know of the accessibility of the system to the printer.

Brief Summary Text (31):
In object linking and embedding processes, an object can be any user-selected group of data, such as a block of text, a set of spreadsheet cells, a chart, sounds, or a graphical image. These data can be embedded in or linked to another document created by a different application. For example, a folder may represent a directory and may contain a group of files, or it may represent a group of programs. Folders can also contain other folders.

Brief Summary Text (33):
As previously stated, the system of the present invention may be used to encrypt and embed objects. The encrypted objects are then not accessible to someone not having a proper key or cryptographic engine. Encrypted embedded objects are not even known to those who do not have access to the container object. Access to the file is further restricted through the use of labels. A label is a field of characters attached to the encrypted file. The label may define a group of people that may have access to the file. The label may define the device at which the file may be accessed. The device may define a single person who may have access. A label may also define the type of access, that is, read only, write only, read and write, print only, etc., that authorized persons may have. A label may also define any combination of authorized people, devices, objects, and/or access type. Thus, the label is a flexible, powerful way to set forth with great specificity all conditions that must be fulfilled in order to gain the defined access to the file. Multiple labels may be attached to an encrypted object. The maximum number of labels that may be used with an object is limited only by the capabilities of the host computer. The system of the present invention is contemplated for use with N labels per object, that is, with any number of labels deemed useful or necessary by the user.

Brief Summary Text (34):
It is important to note that the label is preferably attached to an encrypted object. Thus, an authorized person specified in the label must still have a key to decrypt the object. The system of the present invention can restrict access to an object using labels only, but much stronger protection is available if the object is encrypted as well. Further, passphrase protection is preferably included as a first line of protection. Thus, an authorized user would preferably need three elements to access an object: the correct passphrase, the correct key, and authorization in the label.

Brief Summary Text (36):
The label may appear as a header to authorized users. This header can include pertinent information about the label, such as a list of authorization attributes and a specification of object type. For example, the header may identify the object as a container object, and may further list the objects embedded in the container object, preferably in the form of an array, or tree structure. This form of presentation is preferable because it is more meaningful to a user when the container object contains other container objects as well as other embedded objects, although other header structures, such as lists, may also be utilized. These lower level container objects may also contain further container objects, and so on, for as many levels as needed by the users, limited only by the amount of memory available on the host computer. In the preferred embodiment, users who are not authorized access to the container object will not be able to read the header in decrypted form, and users who are authorized access to the container object but not to one or more objects embedded in the container object will not be able to read the corresponding header array element.

Brief Summary Text (37):
Thus, the header array will show a user all the objects to which he has access. Using this header, the user is allowed random access to any object embedded in the container object at any level, as long as that particular user is authorized at that location. The user simply selects an object embedded at a level lower than the one he has currently accessed, and the system will open all objects along the path between the highest level object and the selected object. Obviously, this will only be allowed if the user has access to all objects along the path. Thus, the user does not have to open the objects along the path one at a time; the system will open them automatically. Of course, the user has the option of opening the objects one at a time if this is desired.

Detailed Description Text (32):
Microsoft.RTM. Object Linking and Embedding (OLE)

Detailed Description Text (37):
Macintosh.RTM. Compound Document Standard

CLAIMS:

1. A method for providing multi-level multimedia security in a data network, comprising:

a) accessing an object-oriented key manager;

b) selecting a first object to encrypt;

c) selecting a first label for the first object;

d) encrypting the first object;

e) labelling the encrypted first object;

f) displaying the first label as a header array;

g) reading the first object label;

h) determining access authorization based on the first object label; and

i) decrypting the first object if access authorization is granted.

4. The method of claim 3, further comprising:

a) reading the header array;

b) determining access authorization based on the second object label; and

c) decrypting the second object if access authorization is granted.

5. The method of claim 4, further comprising presenting a representation of the first object only if access authorization is granted according to the second object label.

8. The method of claim 7, wherein access authorization is print-only authorization.

12. The method of claim 11, further comprising:

a) reading a selected label;

b) determining access authorization to the object associated with the selected label based on the selected label and further based on all labels associated with objects in which the associated object is embedded; and

c) decrypting the associated object if access authorization is granted.

14. A system for providing multi-level multimedia security in a data network, comprising:

A) digital logic means, the digital logic means comprising:

1) a system memory means for storing data;

2) an encryption algorithm module, comprising logic for converting unencrypted objects into encrypted objects, the encryption algorithm module being electronically connected to the system memory means for accessing data stored in the first system memory;

3) means for arranging a plurality of encrypted objects such that at least some of the plurality of encrypted objects are embedded within others of the encrypted objects;

4) an object labelling subsystem, comprising logic means for limiting object access, subject to label conditions of a selected object and further subject to conditions of all labels associated with objects in which the selected object is embedded, the object labelling subsystem being electronically connected to the system memory means for accessing data stored in the system memory means and the object labelling subsystem being further electronically connected to the encryption algorithm module to accept inputs from the encryption algorithm module;

5) a decryption algorithm module, comprising logic for converting encrypted objects into unencrypted objects, the decryption algorithm module being electronically connected to the system memory means for accessing data stored in the system memory

means; and

6) an object label identification subsystem, comprising logic for limiting object access, subject to label conditions, the object label identification subsystem being electronically connected to the system memory means for accessing data stored in the system memory means and the object label identification subsystem being further electronically connected to the decryption algorithm module to accept inputs from the deception algorithm module.

B) the encryption algorithm module working in conjunction with the object labelling subsystem to create an encrypted object such that the object label identification subsystem limits access to an encrypted object.

A not analogous art

Generate Collection | Print

L5: Entry 8 of 10 File: USPT Sep 30, 1997

US-PAT-NO: 5673401
DOCUMENT-IDENTIFIER: US 5673401 A

TITLE: Systems and methods for a customizable sprite-based graphical user interface

DATE-ISSUED: September 30, 1997

INVENTOR-INFORMATION:

NAME	CITY	STATE	ZIP CODE	COUNTRY
Volk; Patrick Michael	Kirkland	WA		
Robin; Michael Breed	Redmond	WA		
Thorne, III; Edwin	Seattle	WA		
Kapell; JoGene	Bellevue	WA		

ASSIGNEE-INFORMATION:

NAME	CITY	STATE	ZIP CODE	COUNTRY	TYPE CODE
Microsoft Corporation	Redmond	WA			02

APPL-NO: 08/ 509083 [PALM]
DATE FILED: July 31, 1995

INT-CL: [06] G06 F 3/00

US-CL-ISSUED: 395/327; 395/356
US-CL-CURRENT: 725/139; 345/763, 345/765, 345/853, 725/131, 725/133, 725/60, 725/61, 725/87

FIELD-OF-SEARCH: 395/155-161, 395/650, 395/700, 395/326-358, 395/701, 345/117-120, 345/145-146, 345/122-131, 348/7, 348/12-13

PRIOR-ART-DISCLOSED:

U.S. PATENT DOCUMENTS

Search Selected | Search ALL

	PAT-NO	ISSUE-DATE	PATENTEE-NAME	US-CL
☐	4656603	April 1987	Dunn	395/159 X
☐	4896291	January 1990	Gest et al.	395/159
☐	5247284	September 1993	Fleming	395/156
☐	5270689	December 1993	Hermann	345/145
☐	5289205	February 1994	Torres	345/125
☐	5295243	March 1994	Robertson et al.	395/160
☐	5303388	April 1994	Kreitman et al.	395/159
☐	5347628	September 1994	Brewer et al.	395/159
☐	5392388	February 1995	Gibson	395/159
☐	5452413	September 1995	Blades	395/159
☐	5479268	December 1995	Young et al.	348/13 X
☐	5485197	January 1996	Hoarty	348/7
☐	5513306	April 1996	Mills et al.	395/352 X
☐	5515486	May 1996	Amro et al.	395/159 X
☐	5517257	May 1996	Dunn et al.	348/7 X
☐	5524195	June 1996	Clanton, III et al.	395/155
☐	5524196	June 1996	Blades	395/155

OTHER PUBLICATIONS

Six et al., "A Software Engineering Perspective to the Design of a User Interface Framework", IEEE, pp. 128-134 1992.
Myers et al., "Garnet: Comprehensive Support for Graphical Highly Interactive user Interfaces", IEEE Computer, pp. 71-85 Nov. 1990.
Advanced Interface Design Guide, IBM, pp. 25-31, 38-39, 46-47, 63-75, 77-81, 83-87 Jun. 1989.
Foley et al., "Computer Graphics: Principles and Practice", Addison-Wesley Pub., pp. 445-451 1990.

ART-UNIT: 245

PRIMARY-EXAMINER: Breene; John E.

ABSTRACT:

An object-oriented system for generating and displaying control items that allow users of an interactive network to recognize and select control functions via a graphical user interface. The manipulation of the control items on a display screen is linked to a set-top terminal associated with the interactive network. The control items, which can be visible or audible, are associated with control objects. Control objects are arranged in a hierarchy, and can contain one or more child control objects. Attributes of a child control object are inherited from an ancestor control object. A control item can be graphically manipulated independently by drawing the control item into its own sprite, or can be manipulated by drawing the control item into the sprite of a parent. The system provides building blocks of control elements that can be composed and customized to produce versatile interfaces for applications and content.

28 Claims, 24 Drawing figures

Generate Collection | Print

L5: Entry 8 of 10 File: USPT Sep 30, 1997

US-PAT-NO: 5673401
DOCUMENT-IDENTIFIER: US 5673401 A

TITLE: Systems and methods for a customizable sprite-based graphical user interface

DATE-ISSUED: September 30, 1997

INVENTOR-INFORMATION:

NAME	CITY	STATE	ZIP CODE	COUNTRY
Volk; Patrick Michael	Kirkland	WA		
Robin; Michael Breed	Redmond	WA		
Thorne, III; Edwin	Seattle	WA		
Kapell; JoGene	Bellevue	WA		

ASSIGNEE-INFORMATION:

NAME	CITY	STATE	ZIP CODE	COUNTRY	TYPE CODE
Microsoft Corporation	Redmond	WA			02

APPL-NO: 08/ 509083 [PALM]
DATE FILED: July 31, 1995

INT-CL: [06] G06 F 3/00

US-CL-ISSUED: 395/327; 395/356
US-CL-CURRENT: 725/139; 345/763, 345/765, 345/853, 725/131, 725/133, 725/60, 725/61, 725/87

FIELD-OF-SEARCH: 395/155-161, 395/650, 395/700, 395/326-358, 395/701, 345/117-120, 345/145-146, 345/122-131, 348/7, 348/12-13

PRIOR-ART-DISCLOSED:

U.S. PATENT DOCUMENTS

Search Selected | Search ALL

	PAT-NO	ISSUE-DATE	PATENTEE-NA	US-CL
☐	4656603	April 1987	Dunn	395/159 X
☐	4896291	January 1990	Gest et al.	395/159
☐	5247284	September 1993	Fleming	395/156
☐	5270689	December 1993	Hermann	345/145
☐	5289205	February 1994	Torres	345/125
☐	5295243	March 1994	Robertson et al.	395/160
☐	5303388	April 1994	Kreitman et al.	395/159
☐	5347628	September 1994	Brewer et al.	395/159
☐	5392388	February 1995	Gibson	395/159
☐	5452413	September 1995	Blades	395/159
☐	5479268	December 1995	Young et al.	348/13 X
☐	5485197	January 1996	Hoarty	348/7
☐	5513306	April 1996	Mills et al.	395/352 X
☐	5515486	May 1996	Amro et al.	395/159 X
☐	5517257	May 1996	Dunn et al.	348/7 X
☐	5524195	June 1996	Clanton, III et al.	395/155
☐	5524196	June 1996	Blades	395/155

OTHER PUBLICATIONS

Six et al., "A Software Engineering Perspective to the Design of a User Interface Framework", IEEE, pp. 128-134 1992.
Myers et al., "Garnet: Comprehensive Support for Graphical Highly Interactive user Interfaces", IEEE Computer, pp. 71-85 Nov. 1990.
Advanced Interface Design Guide, IBM, pp. 25-31, 38-39, 46-47, 63-75, 77-81, 83-87 Jun. 1989.
Foley et al., "Computer Graphics: Principles and Practice", Addison-Wesley Pub., pp. 445-451 1990.

ART-UNIT: 245

PRIMARY-EXAMINER: Breene; John E.

ABSTRACT:

An object-oriented system for generating and displaying control items that allow users of an interactive network to recognize and select control functions via a graphical user interface. The manipulation of the control items on a display screen is linked to a set-top terminal associated with the interactive network. The control items, which can be visible or audible, are associated with control objects. Control objects are arranged in a hierarchy, and can contain one or more child control objects. Attributes of a child control object are inherited from an ancestor control object. A control item can be graphically manipulated independently by drawing the control item into its own sprite, or can be manipulated by drawing the control item into the sprite of a parent. The system provides building blocks of control elements that can be composed and customized to produce versatile interfaces for applications and content.

28 Claims, 24 Drawing figures

L5: Entry 8 of 10 File: USPT Sep 30, 1997

DOCUMENT-IDENTIFIER: US 5673401 A
TITLE: Systems and methods for a customizable sprite-based graphical user interface

DATE FILED (1):
19950731

Abstract Text (1):
An object-oriented system for generating and displaying control items that allow users of an interactive network to recognize and select control functions via a graphical user interface. The manipulation of the control items on a display screen is linked to a set-top terminal associated with the interactive network. The control items, which can be visible or audible, are associated with control objects. Control objects are arranged in a hierarchy, and can contain one or more child control objects. Attributes of a child control object are inherited from an ancestor control object. A control item can be graphically manipulated independently by drawing the control item into its own sprite, or can be manipulated by drawing the control item into the sprite of a parent. The system provides building blocks of control elements that can be composed and customized to produce versatile interfaces for applications and content.

Detailed Description Text (20):
Turning now to the drawings, in which like numerals indicate like elements throughout the several figures, FIG. 1 illustrates the operating environment for an interactive network system. Referring to FIG. 1, the interactive network system 10 includes a headend system 12 for delivering programming information to and receiving instructions from a consumer system 14 via a "two-way" distribution network 16. The headend system 12 is the control center for collecting, organizing, and distributing the signals for all interactive network operations and the source for all programming information. The distribution network 16 transports signals carrying programming information and instructions between the headend system 12 and the consumer system 14. The distribution network 16 can include a world-wide public asynchronous transfer mode (ATM) compatible network with links to the Internet, third party service providers, and other wired and wireless communications networks. The consumer system 14 includes the equipment required for a consumer to receive programming information directly at his or her office or residence and to transmit requests and instructions to the headend system 12.

Detailed Description Text (21):
The headend system 12 can include a set of headend servers 20, including a continuous media server (CMS) system 22 and one or more administrative servers 24, to support various network functions, and a control network 26 linking these headend servers. The headend servers 20 can execute program modules, including service and application program software, to support the transmission of programming information and the reception of requests for such programming information.

Detailed Description Text (22):
It will be appreciated that the headend servers 20 are not necessarily located in one physical location, but can be linked by wired and/or wireless communications paths supplied by the control network. The control network 26 can be a local area network, a wide area network, or a combination of both types of networks. For the preferred embodiment, the control network 26 is implemented as an ATM-based network for routing digital data between the headend servers 20 and the distribution network 16.

Detailed Description Text (25):

To support the [redacted]ks of updating or revising pro[redacted]mming information stored on a memory storage device 30 of the CMS system 22, a computer workstation 32 and a remote server 34 can be connected to the control network 26 via a communications link 36. This communications link allows a program distributor or supplier, which typically operates at a location remote from the CMS system 22, to transmit programming information for storage by one or more of the memory storage devices 30 and eventual distribution to consumers via the headend system 12. The communications link 36 can be implemented by either a wireless or wired communications system. For example, the communications link 36 can be constructed as a microwave link or as a conventional telephone link.

Detailed Description Text (26):
The administrative servers 24 of the headend system 12 can support a variety of services and applications associated with the interactive network system 10, including network security, monitoring, data object storage, financial transactions, data management, and other administrative functions. The administrative servers 24 also handle the interactive service requests or instructions transmitted via the consumer system 14 by consumers. For an application involving a large base of consumers, an administrative server 24 is preferably dedicated to a particular service or function. For example, one or more servers can handle all consumer authorization requirements, whereas other servers can handle network management services, and so forth. Also, administrative servers 24 can be used for data object storage to support network services, such as character-based data associated with VOD services. These data object storage-type servers can support the distribution of video and audio streams by the CMS system 22, and can be implemented as SQL (Structured Query Language) servers.

Detailed Description Text (46):
The set-top terminal 48 can be connected to a peripheral device via input/output (I/O) ports 74. The I/O ports 74 supports the connection of the system bus 70 to a connected peripheral device. For example, the output device 50 can be connected to the I/O ports 74 via a conductor 52. Likewise, an input device 54, such as a game control 90, can be connected to the I/O ports 74. In contrast to the remote control 80, which communicates with the remote control receiver 60 via a wireless communications link, other types of input devices 54 are typically connected to the I/O ports 74 via a cable. Nevertheless, those skilled in the art will appreciate that input devices 54 can communicate with the set-top terminal 48 by use of either wireless or wired communications links.

Detailed Description Text (80):
To change the selection of control items presented by the graphical viewer interface 100, as well as to operate the control functions represented by control items, the user can use the remote control 80 to send command signals to the set-top terminal 48 via an infrared communications link. FIG. 4 shows a front view of a preferred input device for the interactive network system 10, the remote control 80. Referring now to FIGS. 2, 3, and 4, a viewer can manipulate the graphical viewer interface 100 by using the input device to make and communicate selections to the interactive network system 10. The primary input device is the remote control 80, which is the viewer's physical interface to the services offered by the interactive network system 10. The top face section of the remote control 80 includes numerous keys or buttons that allow the user to input commands for controlling functions of the set-top terminal 48 and/or the output device 50.

Detailed Description Text (136):
Graphical user interfaces for conventional computer systems, also described as GUI, are well known in the computer environment. However, those skilled in the art will appreciate that there are necessarily a number of differences between a conventional graphical user interface and a graphical viewer interface for the preferred interactive network system 10. For example, the preferred graphical viewer interface 100 may not include certain conventional elements of a graphical user interface for computers because they do not readily transfer to the environment of the interactive network. These conventional computer GUI elements may include: system menus, menu bars, iconized windows, an user interface for changing window size or position, window housekeeping messages, system level dialog box support, data dynamic data exchange, advanced object linking and embedding features (e.g., compound documents), a clipboard, or single-click and double-click mouse input capabilities.

Detailed Description Text (163):
For the preferred implementation of the graphical viewer interface 100, the frame is

the primary con ner element with which complex cus items and control items are built. Frames ar. analogous to windows in a conventional computer graphical user interface, in that frames are used for displaying and containing groups of other items. Frames can contain graphic images, animation, video, text, and other standard elements, as well as other frames. Typically, the application specifies the content for a frame. Frames can be positioned in absolute screen coordinates or relative to parent frame. For example, a focus frame may be linked to a control frame so that the focus item is always displayed in the same manner relative to the control item. Thus, if the control item is moved, the linked focus item will move with it.

Detailed Description Text (177):
The Control Manager 216 handles the linking and layering of focus objects 208 and control objects 210. A focus object 208 may be linked to a control object 210, which then becomes a parent container holding the focus object 208. The Control Manager 216 maintains a "master" container object. Top-level frames (i.e., frames with no parent) are preferably registered with this master container object. This allows the Control Manager 216 to support the tasks of switching between various program modules 200 when more than one program module is running on the set-top terminal 48.

Detailed Description Text (197):
After completing this initial registration process and presenting an initial display of the graphical viewer interface 100, the set-top terminal 48 is now ready to accept a command from the input device 54. At step 1210, the system receives a command signal in response to the user's decision to conduct a desired control operation. For many control operations, the remote control 80 will be used to transmit an infrared command signal to the set-top terminal 48. The command signal generally affects the control item that currently has focus, i.e., the control item designated as having focus as a result of placing the focus item proximate to this control item.

Neg.
3/13/02

WEST

Help | Logout | Interrupt

Main Menu | Search Form | Posting Counts | Show S Numbers | Edit S Numbers | Preferences | Cases

Search Results -

Terms	Documents
(OLE or (object$ with link$ with embed$)) and author$ and (compound$ adj (document$ or file or article)) and @pd<=19970610	0

Database: US Patents Full-Text Database
US Pre-Grant Publication Full-Text Database
JPO Abstracts Database
EPO Abstracts Database
Derwent World Patents Index
IBM Technical Disclosure Bulletins

Search: L6

Refine Search

Recall Text | Clear

Search History

DATE: Tuesday, March 11, 2003 Printable Copy Create Case

Set Name side by side	Query	Hit Count	Set Name result set
	DB=JPAB,EPAB,DWPI,TDBD; THES=ASSIGNEE; PLUR=YES; OP=OR		
L6	(OLE or (object$ with link$ with embed$)) and author$ and (compound$ adj (document$ or file or article)) and @pd<=19970610	0	L6
	DB=USPT; THES=ASSIGNEE; PLUR=YES; OP=OR		
L5	L4 and link$	10	L5
L4	L3 and (input$ with (authori$ or allow$ or confirm$ or approv$ or accept$))	10	L4
L3	(OLE or (object$ with link$ with embed$)) and author$ and (compound$ adj (document$ or file or article)) and @ad<=19970610	40	L3
L2	(OLE or (object$ with link$ with embed$)) and author$ and @ad<=19970610	325	L2
L1	(OLE or (object$ with link$ with embed$)) and author$ and (compound$ adj data) and @ad<=19970610	1	L1

reviewed

reviewed

END OF SEARCH HISTORY

cited

A

not analogous art

Generate Collection | Print

L5: Entry 1 of 10 File: USPT Feb 16, 1999

US-PAT-NO: 5872926
DOCUMENT-IDENTIFIER: US 5872926 A

TITLE: Integrated message system

DATE-ISSUED: February 16, 1999

INVENTOR-INFORMATION:

NAME	CITY	STATE	ZIP CODE	COUNTRY
Levac; Ronald A.	Hartland	WI		
Bilgrien; Stephen D.	Iron Ridge	WI		
Peters; Michael J.	Menomonee Falls	WI		
Kuecherer; Robert K.	Menomonee Falls	WI		

ASSIGNEE-INFORMATION:

NAME	CITY	STATE	ZIP CODE	COUNTRY	TYPE CODE
Adaptive Micro Systems, Inc.	Milwaukee	WI			02

APPL-NO: 08/ 656377 [PALM]
DATE FILED: May 31, 1996

INT-CL: [06] H04 M 11/00

US-CL-ISSUED: 395/200.36; 340/825.44, 379/93.15, 379/93.24, 379/100.08, 379/100.13, 455/417
US-CL-CURRENT: 709/206; 379/100.08, 379/100.13, 379/93.15, 379/93.24, 455/417

FIELD-OF-SEARCH: 395/200.36, 395/200.68, 395/200.73, 395/200.75, 379/93.01, 379/93.15, 379/93.24, 379/100.01, 379/100.03, 379/100.13, 340/825.44, 455/412, 455/417

PRIOR-ART-DISCLOSED:

U.S. PATENT DOCUMENTS

Search Selected | Search ALL

	PAT-NO	[redacted]-DATE	PATENTEE-N[redacted]	US-CL
☐	4837798	June 1989	Cohen et al.	379/88
☐	4910765	March 1990	Matsuse et al.	379/100.13
☐	5138653	August 1992	Le Clercq	379/93.24
☐	5146488	September 1992	Okada et al.	379/88
☐	5177680	January 1993	Tsukino et al.	395/751
☐	5333266	July 1994	Boaz et al.	395/200.36
☐	5381527	January 1995	Inniss et al.	395/200.69
☐	5406557	April 1995	Baudoin	370/407
☐	5428663	June 1995	Grimes et al.	340/825.44
☐	5459775	October 1995	Isono et al.	379/93.15
☐	5479408	December 1995	Will	340/825.44
☐	5479411	December 1995	Klein	379/88
☐	5487100	January 1996	Kane	340/825.44
☐	5495234	February 1996	Capp et al.	340/825.44
☐	5509000	April 1996	Oberlander	370/409
☐	5513126	April 1996	Harkins et al.	395/200.36
☐	5524137	June 1996	Rhee	379/100.08
☐	5608786	March 1997	Gordon	379/100.08
☐	5627997	May 1997	Pearson et al.	395/200.36

OTHER PUBLICATIONS

http:// cognisoft.com; Information retrieved from Internet Web site on Sep. 25, 1996 and Sep. 26, 1996.
http://www.globalvillage.com; Information retrieved from Internet Web site on Dec. 10, 1996. (Press Release Dated Mar. 18, 1996).
1996 Global Village Communication; FocalPoint data sheets. (No Date).
http://www.frontec.com; Information retrieved from Internet Web site on Dec. 11, 1996. (Press Release Dated Apr. 29, 1996).
1996 Frontec AMT, Inc., Intelligent Messaging Solutions: It's About Time, pp. 1-10.
1996 Frontec AMT, Inc., The AMTrix System Product Overview, pp. 1-8.
Frontec, The AMTrix System Product Description, Sep. 1, 1996.
Messaging: The Intelligent Way, from Business Systems Magazine, dated Aug. 1996.

ART-UNIT: 274

PRIMARY-EXAMINER: Stamber; Eric W.

ABSTRACT:

A system and method for transmitting a message generated by a message source to diverse communication devices. The types of communication devices to receive the message and their respective sites are selected in response to message parameters contained in the message file. The message is converted to a format appropriate for communicating with selected communication devices and then automatically transmitted to the selected sites.

22 Claims, 5 Drawing figures

L5: Entry 1 of 10 File: USPT Feb 16, 1999

DOCUMENT-IDENTIFIER: US 5872926 A
TITLE: Integrated message system

DATE FILED (1):
19960531

Detailed Description Text (3):
As depicted in FIG. 1, system 10 includes a message interface 12, a message server 14, a communication device interface 16 and a plurality of types of communication devices 18a-n. Message interface 12 preferably includes a user interface 20 and an automated source interface 22. User interface 20 permits a user to generate a message, such as with a text editor or a database, including message parameters providing information about the message, such as the intended recipients. Automated source interface 22 allows messages from automated sources, such as other software programs or hardware devices, to be input into system 10. Messages generated by either user interface 20 or automated source interface 22 are relayed to message server 14 over a Dynamic Data Exchange ("DDE") link or in a message server (".msa") file format. In alternative embodiments, messages can be relayed using different methods, such as a net DDE.

Detailed Description Text (4):
The .msa file format is a standardized file format used to submit communications, regardless the source, to message server 14 and preferably is an Object Linking and Embedding ("OLE") compound file format. The actual message is embedded in the .msa file, which also includes data streams specifying parameters, such as date, time and destination, required to transmit the message to selected communication devices 18a-n. In the preferred embodiment, the OLE file includes data streams providing the following information:

Detailed Description Text (37):
Variable data received in in-box 28 is sent to a variable database 29 which maintains a list of variable names and their corresponding parameters, such as value, update rate and destination. Messages received in in-box 28 are sent to a pending message directory 30 where held until activated. When messages are received directory 30 checks a user profile 32 to verify that the user or automated source that generated the message has been authorized to transmit to communication devices 18a-n specified by the message parameters. Users are required to register with a system administrator who assigns and edits the user rights contained in user profile 32 through a user profile editor 34. Similarly, variable database 29 checks with user profile 32 to verify the message source is authorized to send variables to specified communication devices 18a-n.

Detailed Description Text (48):
Referring now to FIG. 4, data flow through an exemplary device driver 26a-n is illustrated. First, a device driver in-box 64 receives the converted file from protocol converters 24a-n. Next, as shown in a data block 66, device driver 26a-n establishes a communication link to selected communication devices 18a-n by accessing device driver information, such as required data field lengths and initialization strings, contained in device driver profile 50. Based on the accessed information, and as illustrated in a data block 68, device driver 26a-n adds device dependent information to the message, such as headers or trailers. In a data block 70, the prepared communication is then transmitted to selected communication devices 18a-n, and the status of the transmission is verified in a data block 72. In a data block 74, the status information is delivered to protocol converter 24a-n through in-box 54. As described above, in the event the status indicates that the

transmission faile[redacted]otocol converter 24a-n will [redacted]t the communication for another attempt.

No authorized Cited

Generate Collection | Print

L5: Entry 2 of 10 File: USPT Oct 6, 1998

US-PAT-NO: 5818447
DOCUMENT-IDENTIFIER: US 5818447 A

TITLE: System and method for in-place editing of an electronic mail message using a separate program

DATE-ISSUED: October 6, 1998

INVENTOR-INFORMATION:

NAME	CITY	STATE	ZIP CODE	COUNTRY
Wolf; Richard J.	Seattle	WA		
Koppolu; Srinivasa R.	Redmond	WA		
Raman; Suryanarayanan	Redmond	WA		
Rayson; Steven J.	Seattle	WA		

ASSIGNEE-INFORMATION:

NAME	CITY	STATE	ZIP CODE	COUNTRY	TYPE CODE
Microsoft Corporation	Redmond	WA			02

APPL-NO: 08/ 660019 [PALM]
DATE FILED: June 6, 1996

INT-CL: [06] G06 T 1/00

US-CL-ISSUED: 345/335; 395/683, 395/200.36, 707/516, 707/524
US-CL-CURRENT: 345/752; 345/853, 709/206, 715/516, 715/524

FIELD-OF-SEARCH: 395/682, 395/683, 395/200.33, 395/285, 395/200.6, 395/200.61, 345/335, 345/352, 707/516, 707/524, 707/530

PRIOR-ART-DISCLOSED:

U.S. PATENT DOCUMENTS

Search Selected | Search ALL

	PAT-NO	ISSUE-DATE	PATENTEE-NAME	US-CL
☐	5682532	October 1997	Remington et al.	395/683
☐	5682536	October 1997	Atkinson et al.	395/683 X
☐	5706458	January 1998	Koppolu	

OTHER PUBLICATIONS

Oski, "Mac clients, take note: Lotus Notes 4.1 finally arrives", MacWEEK, v10, n21, pp. 23(2), May 1996.
Goulde, "Microsoft adapts to the WWW", Distributed Computing Monitor, v11 n2, pp. 21(5), Feb. 1996.
Jones, "Microsoft readies DocObject; technology will allow document editing in Web

browsers", InfoWor[redacted]18 n18, p. 49, Apr. 1996.
Varhol, "Microsoft Activex will complement, not conquer, Java", Government Computer News, v15 n10, pp. 27 (2), May 1996.
Drews, "MIME interoperability: sharing files via E-mail", Network Computing, v7 n6, pp. 96 (2), Apr. 1996.
"OLE 2 Programmer's Reference", Microsoft Press, 1994, vol. 1, pp. 3-29.
"User's Guide: Lotus cc:MAIL, Release 2", Lotus Development Corporation, 1993, pp. 217-236.
"User's Guide: Lotus Ami Pro, Release 3.0", Lotus Development Corporation, 1993, pp. 429-441.

ART-UNIT: 272

PRIMARY-EXAMINER: Feild; Joseph H.

ABSTRACT:

An email client invokes a DocObject-enabled mail note to display an email message and related features of the user interface. The mail note, which is a DocObject container, creates a DocObject server by invoking a DocObject-enabled word processor. The mail note provides a view port in which the word processor displays and edits the body of the email message. The word processor provides its formatting and editing features in the context of the mail note. OLE menu merging provides both email and word processing interoperability while editing the message. Programming interfaces between the mail note and the word processor allow the mail note to translate message data back and forth between the word processor's format and the format imposed by the email client. This ensures that messages created with the word processor can be read by other email clients.

40 Claims, 15 Drawing figures

L5: Entry 2 of 10 File: USPT Oct 6, 1998

DOCUMENT-IDENTIFIER: US 5818447 A
TITLE: System and method for in-place editing of an electronic mail message using a separate program

DATE FILED (1):
19960606

Abstract Text (1):
An email client invokes a DocObject-enabled mail note to display an email message and related features of the user interface. The mail note, which is a DocObject container, creates a DocObject server by invoking a DocObject-enabled word processor. The mail note provides a view port in which the word processor displays and edits the body of the email message. The word processor provides its formatting and editing features in the context of the mail note. OLE menu merging provides both email and word processing interoperability while editing the message. Programming interfaces between the mail note and the word processor allow the mail note to translate message data back and forth between the word processor's format and the format imposed by the email client. This ensures that messages created with the word processor can be read by other email clients.

Brief Summary Text (8):
Finally, a third approach has been one in which users have decided to use a full power word processor for authoring sophisticated and complex documents, and then use email for distribution. This requires the user to work in the word processing context to create and edit the document. When the document is complete, the user must switch to the email program, create a new message, and include the word processor document as an attachment. Although email is an effective mechanism for transporting documents, handling attachments requires several additional steps on the part of both the sender and the recipient of the message.

Drawing Description Text (4):
FIG. 3 illustrates a compound document in a word processing document frame.

Detailed Description Text (3):
Those skilled in the art will appreciate that an exemplary embodiment of the present invention relies on and incorporates several common features of modem personal computers. In order to provide a sufficient background for an embodiment of the present invention, it is useful to first discuss a variety of topics, including an exemplary operating system, the Object Linking and Embedding (OLE) and Document Object (DocObject) interfaces, and the Messaging Application Programming Interface (MAPI). An exemplary embodiment of the present invention will be described after each of these components is briefly discussed.

Detailed Description Text (25):
The operating system 36 provides a variety of functions, services, and interfaces that allow an application program 37a to easily deal with various types of input/output (I/O). This allows the application program 37a to issue relatively simple function calls that cause the operating system 36 to perform the steps required to accomplish various tasks, such as displaying text on the monitor 31 (FIG. 1) or printing text on an attached printer (not shown). Generally described (with reference to FIG. 2), the application program 37a communicates with the operating system 36 by calling predefined functions provided by the operating system 36. The operating system 36 responds by providing the requested information in a message or by executing the requested task.

Detailed Description Text (26):
The OLE and DocObject Interfaces

Detailed Description Text (27):
The "WINDOWS 95" and "WINDOWS NT" operating systems support Microsoft Corporation's Object Linking and Embedding (OLE) and Document Object (DocObject) interfaces. Both OLE and DocObject support a variety of application programming interfaces (APIs) that simplify the interaction between program modules.

Detailed Description Text (28):
OLE is a technology that enables developers to create extensible application programs that operate across multiple platforms. OLE-enabled applications allow users to manipulate information in an intuitive manner, using an environment that is more "document-centric" and less "application-centric." Users can create compound documents with data, or objects, of different formats, and focus directly on the data rather than on the application programs responsible for the data. The data can be embedded within the document, or linked to it, so that only a reference to the data is stored in the document.

Detailed Description Text (29):
OLE facilitates application integration by defining a set of standard interfaces, which are groupings of semantically-related functions through which one program module accesses the services of another. OLE is an open system in the sense that any application program can provide an implementation of a defined interface and any application program can use it. Application programs can either take advantage of built-in functionality provided by OLE, or add to it or replace it as best suits their needs.

Detailed Description Text (30):
The set of OLE services can be viewed as a two tier hierarchy. The lower level contains infrastructure services. These are basic services that provide the means by which features can be implemented and used. The infrastructure services include interface negotiation, memory management, error and status reporting, interprocess communication, structured storage, and data transfer. The upper level of the OLE service hierarchy provides application features, which are the services that benefit the end user. These include compound document management, in-place activation, programmability, and drag and drop operations.

Detailed Description Text (31):
OLE's interfaces provide the standard for component object interaction. Each interface contains a set of functions that defines a contract between the object implementing the interface and the client using it. The contract includes the name of the interface, the function names, and the parameter names and types. Under this contract, the object must implement all of the functions defined for that interface and the implementations must conform to the contract.

Detailed Description Text (32):
All interface names are prefixed with either "I" or "IOle." Interfaces that begin with "IOle" provide services relating to compound document management. Those that begin with "I" provide services that are more general in nature. For example, IOleObject contains methods used by a client of an embedded or linked compound document object. IOleObject is implemented and used only by applications that participate in compound document management. IDataObject, however, contains methods that are used by all applications. These methods provide the means by which data of any type is transferred.

Detailed Description Text (33):
OLE supports the provision of a "compound document," which is a container object that contains a "linked" object or an "embedded" object. The difference between linked and embedded objects has to do with where the actual source data associated with the object is stored. This affects the object's portability, it method of activation and the size of the compound document.

Detailed Description Text (34):
When an object is linked, the source data continues to reside wherever it was initially created, which may be at another point in the document or in another, document altogether. Only a reference, or link, to the object is kept within the compound document. Linking is efficient and minimizes the size of the compound document. Changes made to the source are automatically reflected in any compound

document that has a [redacted] to the source object. From [redacted] user's point of view, a linked object appears to be wholly contained within the document.

Detailed Description Text (35):
With an embedded object, a copy of the original object is physically stored in the compound document, along with all of the information needed to manage that object. As a result, the object becomes a physical part of the document. A compound document containing an embedded object will be larger than one containing the same objects as links. However, embedding offers advantages that offset the larger storage requirement. For example, compound objects with embedded objects can be transferred to another computer and edited there.

Detailed Description Text (36):
Embedded objects can be edited, or activated in place. This means that all maintenance of the object can be done without leaving the compound document. In order to edit the embedded object, the object must be explicitly activated or opened by performing an action such as double-clicking on the object's icon. This results in the object being displayed in a separate window with the user interface provided by the application program that created the object. The object is said to become in-place active (i.e., it is editable), and UI active (i.e., it displays the user interface associated with the application program that created the embedded object).

Detailed Description Text (37):
OLE includes the concepts of native and foreign frames. A frame is a boundary that bounds or frames a view port, and may include menus, toolbars, status bars and the like. A native frame is a frame produced by the application program that created or is associated with the object whose view is being displayed inside the frame. An example of a native frame is a word processing frame in which a view of a word processing document is displayed. With a foreign frame, the frame is produced by an application program that is not associated with the object whose view is being displayed in the frame. An example of a foreign frame is a word processing document frame in which an embedded spreadsheet object is being displayed. FIG. 3 illustrates a word processing document frame 300. The frame 300 includes text 305 and an embedded graphics object 310.

Detailed Description Text (38):
In summary, OLE allows objects to be embedded in a compound document, which is displayed in a container or frame. Generally, the embedded document is displayed in the container in what is referred to as object view. The container controls the appearance of the page and the layout of headers, footers, end notes, etc. The embedded object has no control over these aspects of the page. The container also controls the amount of space that is allocated to the embedded object for displaying its pictorial representation.

Detailed Description Text (39):
Some of the limitations associated with displaying an embedded object in a compound document are addressed by Microsoft Corporation's Document Object, or DocObject, technology. DocObject is an OLE 2.0 interface built on top of OLE and facilitates displaying an object in a "document view" instead of an "object view." The DocObject interface logically partitions a "view" of a document object from a "frame" in which the document object is displayed. The frame specifies the location and dimensions of a view port to which the document object is to display a view. The document view controls the page model and the dimensions of what is displayed within the view port.

Detailed Description Text (42):
A number of logical components are involved in implementing the DocObject interface. These components follow the client/server model, wherein a client (or DocObject container) requests service from a DocObject server. FIG. 5a illustrates the components that are included on the server side of a DocObject interface. The server side includes a document 500 and a view 505. The document 500 and the view 505 may be included in a single object or may constitute separate objects. The lines and circles that extend from the document 500 and view 505 specify the interfaces that are supported by each logical component. In FIG. 5a, the IMsoDocument and IMsoView interfaces are part of the DocObject interface. The remaining interfaces are part of OLE 2.0.

Detailed Description Text (43):

FIG. 5b illustrates [redacted] logical components that form [redacted] client side of the DocObject interface. The logical components include a frame 550, a document container 555, a document site 560, and a view site 565. The frame 550 provides the frame for the document object. The document container 555 is the container that stores the document object. The document site 560 serves as the site for the embedded document object. The view site 565 provides a site on the client for the view 505 (FIG. 5a). The view in the view site 565 supports interfaces that enable a view 505 to communicate with the document container 555. The interfaces supported by the client side components are illustrated in FIG. 5b. The IMsoDocumentSite interface is part of the DocObject interface. The remaining interfaces are part of OLE 2.0.

Detailed Description Text (44):
Referring still to FIGS. 5a and 5b, the interfaces will be briefly described. The document 500 supports at least three standard OLE interfaces, including IPersistStorage, IDataobject, and IOleObject. The IPersistStorage interface is the interface through which the document container 555 communicates with the document 500 regarding storage. The IDataobject interface allows data to be passed to and from the document 500. The IOleObject interface is the primary interface though which an embedded object provides functionality to its container.

Detailed Description Text (46):
The view 505 supports the OLE's standard IOleInPlaceObject and IOleInPlaceActiveObject interfaces. The IOleInPlaceObject interface is used primarily to allow a container to communicate with its contained object, and includes functions for deactivating the object and its associated user interface. The IOleInPlaceActiveObject interface provides an interface for the top level container to communicate with currently active objects.

Detailed Description Text (48):
Referring still to FIGS. 5a and 5b, the interfaces implemented by the client side components will now be described. The frame 550 supports the standard OLE IOleInPlaceFrame interface. Its functions allow the insertion, removal, and manipulation of menus for the frame. The interface also includes functions for displaying text on a status line or for enabling or disabling modeless dialogs.

Detailed Description Text (49):
The document container 555 supports the standard OLE IOleContainer interface. This interface provides the ability to enumerate objects in a container and to keep a container running in the absence of other reasons for it to continue running.

Detailed Description Text (51):
The document site 560 also supports the standard OLE IOleClientSite interface. This is the primary interface by which an object requests services from its container.

Detailed Description Text (52):
The view site 565 supports the standard OLE IOleInPlaceSite interface. This interface includes a number of functions that allow an in-place active object to communicate with its immediate container. The view 505 communicates with the view site 565 via functions in the IOleInPlaceSite interface.

Detailed Description Text (56):
From the foregoing brief description, it should be appreciated OLE and DocObject, like other APIs, are quite complex and provide a wide variety of services that allow program modules to easily interface with each other. For more comprehensive information regarding OLE and DocObject, the reader may refer to any of a variety of publications, including the "OLE 2 Programmer's Reference," published by Microsoft Press, and the "OLE Document Object Specification," published by Microsoft Corporation.

Detailed Description Text (60):
The programming interfaces are used by the MAPI subsystem 705, by a client application 700, and by service provider writers. The main programming interface is an object-based interface known as the MAPI programming interface, which is based on the OLE Component Object Model.

Detailed Description Text (78):
FIG. 10 illustrates an exemplary embodiment of the present invention. In this embodiment, the email client's default send and read mail notes have been replaced

with DocObject-enabled send and read mail notes 1005. The DocObject-enabled mail note is a DocObject container that provides a view port or frame that allows the DocObject-enabled word processor 1010 (or DocObject server) to display and edit an email message in the view port provided by the mail note 1005. In other words, the mail note 1005 and word processor 1010 utilize the OLE and DocObject interfaces to allow the word processor 1010 to display a document object in a view port provided by the mail note 1005. In addition to the OLE and DocObject interfaces, the preferred embodiment of the present invention defines three new interfaces, including IMsoMailSite, IMsoFormSite, and IMsoMailEditor, which are described below. In order to distinguish a DocObject-enabled mail note 1005 from the standard email client mail note 905, the DocObject-enabled mail note will be referred to as a "container mail note." Although the send and receive container mail notes replace the email clients standard mail notes, none of the other installed forms are affected. An example of a suitable DocObject-enabled word processor 1010 is version 7.0 of the "MICROSOFT WORD" application program.

Detailed Description Text (79):
The container mail note continues to provide the user with email-related commands and functionality. The container mail note provides the MAPI functionality associated with handling the disposition of messages (e.g., send, reply, forward, etc.) and the exchange of data between the MAPI message store and the container mail note. The word processor, which is the DocObject server, provides word processing commands and functionality and exchanges data with the container mail note. The email and word processing commands displayed to the user are merged via the OLE menu merging functions. This provides the user with immediate access to menus and toolbars associated with both the email client and the word processor.

Detailed Description Text (80):
Although using a full powered word processor in the context of a container mail note provides improved editing and formatting capabilities, those skilled in the art will understand that messages created in the word processor's native format must be compatible with a variety of email clients. The message format is defined by the email client. In an exemplary embodiment, the email client complies with the MAPI specification. The MAPI format ensures the interoperability between an embodiment of the present invention, a prior art rich text mail client, and other mail clients and gateways. This is particularly important with respect to an attachment. For example, a word processor may create a compound document in which the attached files are embedded in the document. This type of attachment handling is incompatible with MAPI, which requires that attachments be stored separately at the end of the message.

Detailed Description Text (86):
The OLE and DocObject interfaces described above allow the word processor 1010 to display text in a view port provided by the container mail note 1005. However, in order to allow the container mail note 1005 to transport data between the MAPI message store and the word processor 1010, it is necessary to define a protocol for transferring data between the container mail note and the word processor. This protocol must ensure that attachments are handled in a way that complies with MAPI.

Detailed Description Text (87):
In addition to the OLE and DocObject interfaces discussed earlier, the preferred embodiment of the present invention employs three additional interfaces in order to facilitate communication between the container mail note 1005 and the word processor 1010. The container mail note 1005, which is the DocObject container, implements the IMsoMailSite and IMsoFormSite interfaces. The word processor 1010, which is the DocObject server, implements the IMsoMailEditor interface. Each of these interfaces and their associated functions are described below.

Detailed Description Text (89):
This member function is used by the word processor 1010 when the user inserts an attachment into the message. The word processor 1010 passes the container mail note 1005 the name of the attachment file and the container mail note returns an MSOOBJECT that can be embedded in the message at the current character position. An MSOOBJECT is an OLE object that is embeddable by the word processor's OLE container support.

Detailed Description Text (90):
This member function is used by the word processor 1010 to convert between an MSOOBJECT that is a mail attachment and an MSOOBJECT that is a Packager OLE object

that contains the a[redacted]ed file. This is used when t[redacted]er moves an MSOOBJECT from a container mail note 1005 to a non-mail document (such as a spreadsheet or a regular word processor document) or the reverse. MSOOBJECT's that are mail attachments can only exist in a container mail note.

Detailed Description Text (95):
This member function is used by the container mail note 1005 to set and clear the contents of the message. The container mail note passes the word processor a group of flags, a format (cfText or cfRTF), an IStream, the author name (for revision marking), and a list of MSOOBJECTS with their corresponding character positions in the message. The group of flags tells the word processor whether to set or clear the message, whether its a New, Read or Compose message etc. The IStream contains the contents of the message in the specified cf (Windows clipboard) format.

Detailed Description Text (108):
At step 1245 the selected mail item is displayed on the monitor. The container mail note provides a DocObject container that includes a menu bar, toolbars, and header information. The view port provided by the DocObject container displays the body of the message under the control of the word processor. Those skilled in the art will appreciate that the displayed window includes both email and word processing commands and buttons as a result of OLE menu merging. This allows the user to perform email and word processing related functions without having to change contexts.

Detailed Description Text (125):
From the foregoing description, it will be appreciated that the present invention provides an improved system and method for editing email messages using a full powered word processor. An exemplary embodiment of the present invention is embodied in a container mail note and word processing program that implement the OLE and DocObject interfaces to allow the word processor to display the email message in a view port provided by the container mail note. Additional interfaces are implemented to allow the container mail note to translate data between the word processor and an email client.

Other Reference Publication (6):
"OLE 2 Programmer's Reference", Microsoft Press, 1994, vol. 1, pp. 3-29.

NO

Generate Collection | Print

L5: Entry 3 of 10 File: USPT Jun 9, 1998

US-PAT-NO: 5765152
DOCUMENT-IDENTIFIER: US 5765152 A

TITLE: System and method for managing copyrighted electronic media

DATE-ISSUED: June 9, 1998

INVENTOR-INFORMATION:

NAME	CITY	STATE	ZIP CODE	COUNTRY
Erickson; John S.	Norwich	VT		

ASSIGNEE-INFORMATION:

NAME	CITY	STATE	ZIP CODE	COUNTRY	TYPE CODE
Trustees of Dartmouth College	Hanover	NH			02

APPL-NO: 08/ 543161 [PALM]
DATE FILED: October 13, 1995

INT-CL: [06] G06 F 17/30

US-CL-ISSUED: 707/9; 707/1
US-CL-CURRENT: 707/9; 707/1

FIELD-OF-SEARCH: 395/600, 707/9, 701/1

PRIOR-ART-DISCLOSED:

U.S. PATENT DOCUMENTS

Search Selected | Search ALL

	PAT-NO	[redacted] DATE	PATENTEE-N[redacted]	US-CL
☐	4218738	August 1980	Matyas et al.	364/200
☐	4528643	July 1985	Freeny	364/900
☐	4827508	May 1989	Shear	380/4
☐	4888798	December 1989	Earnest	380/4
☐	4941175	July 1990	Enescu et al.	380/4
☐	4977594	December 1990	Shear	380/4
☐	4999806	March 1991	Chernow et al.	364/900
☐	5023907	June 1991	Johnson et al.	380/4
☐	5050213	September 1991	Shear	380/25
☐	5113518	May 1992	Durst et al.	395/550
☐	5138712	August 1992	Corbin	395/700
☐	5182770	January 1993	Medveczky et al.	380/4
☐	5410598	April 1995	Shear	380/4
☐	5495411	February 1996	Ananda	364/401
☐	5548645	August 1996	Ananda	380/4
☐	5553143	September 1996	Ross et al.	380/25

FOREIGN PATENT DOCUMENTS

FOREIGN-PAT-NO	PUBN-DATE	COUNTRY	US-CL
WO 94/27228	November 1994	WO	

OTHER PUBLICATIONS

Choudhury et al., Copyright Protection for Electronic Publishing Over Computer Networks, 8302 IEEE Network 9, No. 3 (1995).
IBM Technical Disclosure Bulletin vol. 37, No. 03 (1994).

ART-UNIT: 271

PRIMARY-EXAMINER: Black; Thomas G.

ASSISTANT-EXAMINER: Corrielus; Jean M.

ABSTRACT:

Copyrighted electronic media are packaged in a secure electronic format, and registered on associated registration server, which serves to provide on-line licensing and copyright management for that media. Users are connected to the server, e.g., through a computer network or the Internet, to enable data transfers and to transact licenses to utilize the media. Packaged electronic media are typically created by an author or derivative user of the work. Once the packaged media is registered on the server, the media is made available for limited use and possible license through an authorization server. This limited use is specified within the minimum permissions data set assigned to each packaged media. Without a license, users are typically permitted to view the packaged media--through a system which unpackages the media--but cannot save or otherwise transfer the media without obtaining auxiliary permissions to do so from the authorization server. The electronic media is authenticated through digital signatures and optional encryption.

34 Claims, 20 Draw[redacted]igures

[redacted]

```
B 233, 256, 275, 621, 634, 647
       10mar03 10:21:55 User264717 Session D213.2
          $170.71   85.353 DialUnits File411
    $170.71  Estimated cost File411
     $16.56  INTERNET
    $187.27  Estimated cost this search
    $187.97  Estimated total session cost  85.588 DialUnits

SYSTEM:OS  - DIALOG OneSearch
  File 233:Internet & Personal Comp. Abs.  1981-2003/Feb
         (c) 2003 Info. Today Inc.
  File 256:SoftBase:Reviews,Companies&Prods.  82-2003/Jan
         (c)2003 Info.Sources Inc
  File 275:Gale Group Computer DB(TM)  1983-2003/Mar 06
         (c) 2003 The Gale Group
  File 621:Gale Group New Prod.Annou.(R)  1985-2003/Mar 06
         (c) 2003 The Gale Group
  File 634:San Jose Mercury   Jun 1985-2003/Mar 08
         (c) 2003 San Jose Mercury News
  File 647:CMP  Computer Fulltext  1988-2003/Feb W3
         (c) 2003 CMP Media, LLC

      Set  Items  Description
      ---  -----  -----------
?
```

scanned

```
S (AUTHOR? OR APPROV?) AND ((COMPOUND? (W) DOCUMENT) AND (LINK? (W2) (EMBED? (W) OBJ

Your SELECT statement is:
    S (AUTHOR? OR APPROV?) AND ((COMPOUND? (W) DOCUMENT) AND (LINK? (W2)
(EMBED? (W) OBJECT)) OR OLE) AND OLE AND PD<=970610

            Items   File
            -----   ----
              121     9: Business & Industry(R)_Jul/1994-2003/Mar 07
               53    13: BAMP_2003/Jan W4
>>>File 15 processing for PD=  : PD=970610
>>>File 15:     started at PD=710000 stopped at PD=930107
               62    15: ABI/Inform(R)_1971-2003/Mar 08
>>>File 16 processing for PD=  : PD=970610
>>>File 16:     started at PD=19900101 stopped at PD=19950623
              210    16: Gale Group PROMT(R)_1990-2003/Mar 07
>>>File 18 processing for PD=  : PD=970610
>>>File 18:     started at PD=19860423 stopped at PD=19931110
                2    18: Gale Group F&S Index(R)_1988-2003/Mar 06
                2    73: EMBASE_1974-2003/Mar W1
               33    76: TGG Management Contents(R)_86-2003/Feb W4
                4   112: UBM Industry News_1998-2003/Mar 10
                1   122: Harvard Business Review_1971-2003/Feb
>>>File 129 processing for PD=  : PD=970610
>>>File 129:     started at PD=30126 stopped at PD=930816
                3   129: PHIND(Archival)_1980-2003/Mar W1
>>>File 132 processing for PD=  : PD=970610
>>>File 132:     started at PD=850703 stopped at PD=911118
                1   132: S&P`s  Daily  News_1985-2003/Mar 07
                3   135: NewsRx Weekly Reports_1995-2003/Feb W4
               13   147: The Kansas City Star_1995-2003/Mar 09
>>>File 148 processing for PD=  : PD=970610
>>>File 148:     started at PD=140105 stopped at PD=830728
                7   148: Gale Group Trade & Industry DB_1976-2003/Mar 06
>>>File 149 processing for PD=  : PD=970610
>>>File 149:     started at PD=760100 stopped at PD=830406
                1   149: TGG Health&Wellness DB(SM)_1976-2003/Feb W3
       Examined  50 files
>>>File 180 processing for PD=  : PD=970610
>>>File 180:     started at PD=19850102 stopped at PD=19921224
               10   180: Federal Register_1985-2003/Mar 07
>>>File 194 processing for PD=  : PD=970610
>>>File 194:     started at PD=820913 stopped at PD=900601
                2   194: FBODaily_1982/Dec-2003/Nov
               54   233: Internet & Personal Comp. Abs._1981-2003/Feb
               72   256: SoftBase:Reviews,Companies&Prods._82-2003/Jan
                3   264: DIALOG Defense Newsletters_1989-2003/Mar 07
>>>File 275 processing for PD=  : PD=970610
>>>File 275:     started at PD=140103 stopped at PD=881206
                2   275: Gale Group Computer DB(TM)_1983-2003/Mar 06
>>>File 319 processing for PD=  : PD=970610
>>>File 319:     started at PD=19950130 stopped at PD=900825
                1   319: Chem Bus NewsBase_1984-2003/Mar 05
               13   387: The Denver Post_1994-2003/Mar 07
                8   392: Boston Herald_1995-2003/Mar 07
       Examined 100 files
               12   427: Fort Worth Star-Telegram_1993-2003/Mar 07
                6   433: Charleston Newspapers_1997-2003/Mar 08
>>>File 476 processing for PD=  : PD=970610
>>>File 476:     started at PD=19820102 stopped at PD=19881015
               17   476: Financial Times Fulltext_1982-2003/Mar 10
>>>File 483 processing for PD=  : PD=970610
>>>File 483:     started at PD=861019 stopped at PD=940620
                4   483: Newspaper Abs Daily_1986-2003/Mar 08
>>>File 484 processing for PD=  : PD=970610
```

```
>>>File 484:      started at PD=860000 stopped at PD=910622
                3   484: Periodical Abs Plustext_1986-2003/Mar W1
>>>File 485 processing for PD=  : PD=970610
>>>File 485:      started at PD=130000 stopped at PD=920202
                1   485: Accounting & Tax DB_1971-2003/Mar W1
               37   486:  Press-Telegram_1992- 2003/Mar 07
               10   487: Columbus Ledger-Enquirer_1994-2003/Mar 08
                2   488: Duluth News-Tribune_1995-2003/Mar 07
>>>File 489 processing for PD=  : PD=970610
>>>File 489:      started at PD=900806 stopped at PD=961228
                5   489: The News-Sentinel_1991-2003/Mar 07
                6   490: Tallahassee Democrat_1993- 2003/Feb 14
>>>File 492 processing for PD=  : PD=970610
>>>File 492:      started at PD=11/10/99 stopped at PD=910923
               52   492: Arizona Repub/Phoenix Gaz_19862002/Jan 06
>>>File 494 processing for PD=  : PD=970610
>>>File 494:      started at PD=2/7/2001 stopped at PD=930611
               62   494: St LouisPost-Dispatch_1988-2003/Mar 09
>>>File 498 processing for PD=  : PD=970610
>>>File 498:      started at PD=12 stopped at PD=920619
               19   498: Detroit Free Press_1987-2003/Mar 07
                2   501: Extel Intl News Cards_1995-2002/Mar W4
                1   512: ESPICOM Telecom./Power Rpts_2002/Jun
        Examined 150 files
                8   532: Bangor Daily News_1996-2003/Mar 08
                5   536: (GARY) POST-TRIBUNE_1992-1999/Dec 30
                3   538: Boca Raton News_1994- 1999/Jul 05
                4   539: Macon Telegraph_1994-2003/Mar 06
               28   541: SEC Online(TM) Annual Repts_1997/Sep W3
               75   542: SEC Online(TM) 10-K Reports_1997/Sep W3
               16   543: SEC Online(TM) 10-Q Reports_1997/Sep W3
>>>File 544 processing for PD=  : PD=970610
>>>File 544:      started at PD=860811 stopped at PD=941018
               19   544: SEC Online(TM)  Proxy Repts_1997/Sep W3
>>>File 545 processing for PD=  : PD=970610
>>>File 545:      started at PD=820101 stopped at PD=890330
                1   545: Investext(R)_1982-2003/Mar 10
>>>File 554 processing for PD=  : PD=970610
>>>File 554:      started at PD=19900101 stopped at PD=19951213
                1   554: TFSD J V & Alliances_1990-2003/Mar 10
                1   560: Spokane Spokesman-Review_1994-2003/Mar 07
                4   563: Key Note Market Res._1986-2001/Aug 03
               18   564: ICC Brit.Co.Ann.Rpts_1984-2003/Mar 09
>>>File 570 processing for PD=  : PD=970610
>>>File 570:      started at PD=19840102 stopped at PD=19910623
                2   570: Gale Group MARS(R)_1984-2003/Mar 06
                2   576: Aberdeen American News_1995-2003/Mar 04
               24   577: Roanoke Times_1992-2003/Mar 07
        Examined 200 files
                5   582: Augusta Chronicle_1996- 2003/Mar 09
                4   587: Jane`s Defense&Aerospace_2003/Mar W1
                1   589: FI Defense Market Intelligence_2003/Mar 03
>>>File 608 processing for PD=  : PD=970610
>>>File 608:      started at PD=108 stopped at PD=970111
               38   608: KR/T Bus.News._1992-2003/Mar 10
               14   619: Asia Intelligence Wire_1995-2003/Mar 09
                3   620: EIU:Viewswire_2003/Mar 08
>>>File 621 processing for PD=  : PD=970610
>>>File 621:      started at PD=00000000 stopped at PD=19910208
                5   621: Gale Group New Prod.Annou.(R)_1985-2003/Mar 06
                9   623: Business Week_1985-2003/Mar 08
>>>File 624 processing for PD=  : PD=970610
>>>File 624:      started at PD=104 stopped at PD=921201
               16   624: McGraw-Hill Publications_1985-2003/Mar 08
>>>File 625 processing for PD=  : PD=970610
```

```
>>>File 625:      started at PD=8111 stopped at PD=890425
                1   625: American Banker Publications_1981-2003/Mar 10
>>>File 626 processing for PD=  : PD=970610
>>>File 626:      started at PD=8111 stopped at PD=890601
                1   626: Bond Buyer Full Text_1981-2003/Mar 10
               34   627: EIU: Country Analysis_2003/Feb W4
                5   628: Ctry Risk & Forecasts_2003/Feb W4
                2   629: EIU:BUS. Newsletters_2003/Feb W4
>>>File 631 processing for PD=  : PD=970610
>>>File 631:      started at PD=11/14/99 stopped at PD=850519
               51   631: Boston Globe_1980-2003/Mar 09
>>>File 633 processing for PD=  : PD=970610
>>>File 633:      started at PD=830101 stopped at PD=880823
               49   633: Phil.Inquirer_1983-2003/Mar 05
>>>File 634 processing for PD=  : PD=970610
>>>File 634:      started at PD=850602 stopped at PD=901209
               21   634: San Jose Mercury_ Jun 1985-2003/Mar 08
>>>File 635 processing for PD=  : PD=970610
>>>File 635:      started at PD=1190 stopped at PD=910826
               58   635: Business Dateline(R)_1985-2003/Mar 08
>>>File 636 processing for PD=  : PD=970610
>>>File 636:      started at PD=19880101 stopped at PD=19940316
               87   636: Gale Group Newsletter DB(TM)_1987-2003/Mar 06
>>>File 637 processing for PD=  : PD=970610
>>>File 637:      started at PD=1986 stopped at PD=940608
               41   637: Journal of Commerce_1986-2003/Mar 10
>>>File 638 processing for PD=  : PD=970610
>>>File 638:      started at PD=25, stopped at PD=920701
               46   638: Newsday/New York Newsday_1987-2003/Mar 09
>>>File 640 processing for PD=  : PD=970610
>>>File 640:      started at PD=850209 stopped at PD=930621
               31   640: San Francisco Chronicle_1988-2003/Mar 09
>>>File 641 processing for PD=  : PD=970610
>>>File 641:      started at PD=890523 stopped at PD=941109
               66   641: Rocky Mountain News_Jun 1989-2003/Mar 08
>>>File 642 processing for PD=  : PD=970610
>>>File 642:      started at PD=11/04/98 stopped at PD=930610
               43   642: The Charlotte Observer_1988-2003/Mar 09
                8   643: Grand Forks Herald_1995-2003/Mar 09
                4   644: (Boulder)  Daily Camera_1995-2003/Mar 07
       Examined 250 files
               11   645: Contra Costa Papers_1995- 2003/Mar 08
              320   647: CMP  Computer Fulltext_1988-2003/Feb W3
                3   648: TV and Radio Transcripts_1997-2003/Mar W1
>>>File 649 processing for PD=  : PD=970610
>>>File 649:      started at PD=830104 stopped at PD=891231
               15   649: Gale Group Newswire ASAP(TM)_2003/Mar 06
>>>File 660 processing for PD=  : PD=970610
>>>File 660:      started at PD=901001 stopped at PD=960721
               26   660: Federal News Service_1991-2002/Jul 02
                1   667: ITAR/TASS News_1996-1999/May 26
               23   696: DIALOG Telecom. Newsletters_1995-2003/Mar 08
>>>File 701 processing for PD=  : PD=970610
>>>File 701:      started at PD=5/12/00 stopped at PD=930929
               25   701: St Paul Pioneer Pr Apr_1988-2003/Mar 07
>>>File 702 processing for PD=  : PD=970610
>>>File 702:      started at PD=801018 stopped at PD=880603
               58   702: Miami Herald_1983-2003/Mar 07
>>>File 703 processing for PD=  : PD=970610
>>>File 703:      started at PD=880531 stopped at PD=951205
               58   703: USA Today_1989-2003/Mar 07
>>>File 704 processing for PD=  : PD=970610
>>>File 704:      started at PD=890101 stopped at PD=940624
               46   704: (Portland)The Oregonian_1989-2003/Mar 08
>>>File 706 processing for PD=  : PD=970610
```

```
>>>File 706:      started at PD=10100 stopped at PD=940611
              267   706: (New Orleans)Times Picayune_1989-2003/Mar 08
>>>File 707 processing for PD=  : PD=970610
>>>File 707:      started at PD=19333333 stopped at PD=940510
               36   707: The Seattle Times_1989-2003/Mar 09
>>>File 708 processing for PD=  : PD=970610
>>>File 708:      started at PD=6XSECTION: stopped at PD=940618
               13   708: Akron Beacon Journal_1989-2003/Mar 06
>>>File 709 processing for PD=  : PD=970610
>>>File 709:      started at PD=14 stopped at PD=940618
               35   709: Richmond Times-Disp._1989-2003/Mar 08
>>>File 710 processing for PD=  : PD=970610
>>>File 710:      started at PD=880601 stopped at PD=931205
               30   710: Times/Sun.Times(London)_Jun 1988-2003/Mar 10
>>>File 711 processing for PD=  : PD=970610
>>>File 711:      started at PD=880919 stopped at PD=941117
               41   711: Independent(London)_Sep 1988-2003/Mar 08
>>>File 712 processing for PD=  : PD=970610
>>>File 712:      started at PD=11 stopped at PD=940616
               51   712: Palm Beach Post_1989-2003/Mar 08
>>>File 713 processing for PD=  : PD=970610
>>>File 713:      started at PD=880410 stopped at PD=940606
              146   713: Atlanta J/Const._1989-2003/Mar 09
>>>File 714 processing for PD=  : PD=970610
>>>File 714:      started at PD=900903 stopped at PD=960305
               71   714: (Baltimore) The Sun_1990-2003/Mar 08
               10   715: Christian Sci.Mon._1989-2003/Mar 10
>>>File 716 processing for PD=  : PD=970610
>>>File 716:      started at PD=881210 stopped at PD=940622
               31   716: Daily News Of L.A._1989-2003/Mar 07
>>>File 717 processing for PD=  : PD=970610
>>>File 717:      started at PD=890101 stopped at PD=950715
               68   717: The Washington Times_Jun 1989-2003/Mar 07
>>>File 718 processing for PD=  : PD=970610
>>>File 718:      started at PD=22 stopped at PD=960411
               26   718: Pittsburgh Post-Gazette_Jun 1990-2003/Mar 10
>>>File 719 processing for PD=  : PD=970610
>>>File 719:      started at PD=400100 stopped at PD=910819
               16   719: (Albany) The Times Union_Mar 1986-2003/Mar 07
>>>File 720 processing for PD=  : PD=970610
>>>File 720:      started at PD=18 stopped at PD=930521
                5   720: (Columbia) The State_Dec 1987-2003/Mar 09
>>>File 721 processing for PD=  : PD=970610
>>>File 721:      started at PD=11/11/1999 stopped at PD=950524
               66   721: Lexington Hrld.-Ldr._1990-2003/Mar 07
>>>File 722 processing for PD=  : PD=970610
>>>File 722:      started at PD=900327 stopped at PD=960815
               38   722: Cincinnati/Kentucky Post_1990-2003/Mar 06
>>>File 723 processing for PD=  : PD=970610
>>>File 723:      started at PD=4/13/98 stopped at PD=950622
               21   723: The Wichita Eagle_1990-2003/Mar 07
>>>File 724 processing for PD=  : PD=970610
>>>File 724:      started at PD=890101 stopped at PD=940623
               76   724: (Minneapolis)Star Tribune_1989-1996/Feb 04
       Examined 300 files
                5   726: S.China Morn.Post_1992--2003/Mar 07
Processing
>>>File 727 processing for PD=  : PD=970610
>>>File 727:      started at PD=107280 stopped at PD=950526
              245   727: Canadian Newspapers_1990-2003/Mar 10
               21   728: Asia/Pac News_1994-2003/Mar W1
>>>File 731 processing for PD=  : PD=970610
>>>File 731:      started at PD=65BRUARY stopped at PD=890821
               13   731: Philad.Dly.News_1983- 2003/Mar 05
>>>File 732 processing for PD=  : PD=970610
```

```
>>>File 732:      started at PD=900611 stopped at PD=960213
              33   732: San Francisco Exam._1990- 2000/Nov 21
>>>File 733 processing for PD=  : PD=970610
>>>File 733:      started at PD=15 stopped at PD=950617
              25   733: The Buffalo News_1990- 2003/Mar 08
>>>File 734 processing for PD=  : PD=970610
>>>File 734:      started at PD=4 2002 stopped at PD=960316
              38   734: Dayton Daily News_Oct 1990- 2003/Mar 07
>>>File 735 processing for PD=  : PD=970610
>>>File 735:      started at PD=25UARY stopped at PD=950407
              75   735: St. Petersburg Times_1989- 2000/Nov 01
>>>File 736 processing for PD=  : PD=970610
>>>File 736:      started at PD=891019 stopped at PD=950620
              23   736: Seattle Post-Int._1990-2003/Mar 08
>>>File 738 processing for PD=  : PD=970610
>>>File 738:      started at PD=900101 stopped at PD=950627
              18   738: (Allentown) The Morning Call_1990-2003/Mar 08
>>>File 739 processing for PD=  : PD=970610
>>>File 739:      started at PD=900101 stopped at PD=950628
              35   739: The Fresno Bee_1990-2003/Mar 07
>>>File 740 processing for PD=  : PD=970610
>>>File 740:      started at PD=900627 stopped at PD=951217
             517   740: (Memphis)Comm.Appeal_1990-2003/Mar 08
>>>File 741 processing for PD=  : PD=970610
>>>File 741:      started at PD=11/30/02 stopped at PD=960217
              39   741: (Norfolk)Led./Pil._1990-2003/Mar 08
>>>File 742 processing for PD=  : PD=970610
>>>File 742:      started at PD=11 stopped at PD=951022
              25   742: (Madison)Cap.Tim/Wi.St.J_1990-2003/Mar 08
>>>File 743 processing for PD=  : PD=970610
>>>File 743:      started at PD=19891300 stopped at PD=940730
              29   743: (New Jersey)The Record_1989-2003/Mar 07
              72   744: (Biloxi) Sun Herald_1995-2003/Mar 02
               7   748: Asia/Pac Bus. Jrnls_1994-2003/Mar 06
               1   754: IPO Maven_1994-2000/Jul
              20   755: New Zealand Newspapers_1995-2003/Mar 09
               1   761: Datamonitor Market Res._1992-2003/Mar
               8   765: Frost & Sullivan_1992-1999/Apr
               2   766: (R)Kalorama Info Market Res._1993-2000/Aug
              22   781: ProQuest Newsstand_1998-2003/Mar 07
               1   788: (Myrtle Beach) The Sun News_1996-2003/Mar 05
>>>File 810 processing for PD=  : PD=970610
>>>File 810:      started at PD=850116 stopped at PD=911127
              17   810: Business Wire_1986-1999/Feb 28
>>>File 813 processing for PD=  : PD=970610
>>>File 813:      started at PD=100000 stopped at PD=900920
               9   813: PR Newswire_1987-1999/Apr 30
       Examined 350 files
               8   816: Canada NewsWire_1996-1999/Jun 24
               2   818: Xinhua News_1996-1999/May 26
               7   861: UPI News_1996-1999/May 27
              18   929: Albuquerque Newspapers_1995-2003/Mar 09
              10   980: Sarasota Herald-Tribune_1996-2003/Mar 09

   143 files have one or more items; file list includes 357 files.
   One or more terms were invalid in 182 files.

?
```